N° d'ordre 07/2005-M/IN

REBUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE D'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE

« HOUARI BOUMEDIENE »

FACULTE D'ELECTRONIQUE ET D'INFORMATIQUE

MEMOIRE

Présenté pour l'obtention du diplôme de MAGISTER

En : INFORMATIQUE

Spécialité : Programmation et Systèmes

Par : ALIOUANE LYNDA

SUJET

L'AUTHENTIFICATION DANS LES RESEAUX AD HOC

Soutenu Le 02/07/2005 , devant le jury composé de :

Mr. A. AISSANI	Professeur, USTHB,	Président
Mr. N. BADACHE	Professeur, USTHB,	Directeur de thèse
Mme Z. ALIMAZIGHI	Professeur, USTHB,	Examinateur
Mr S. LARABI	Maître de conférences, USTHB,	Examinateur

REMERCIEMENT

Je tiens à remercier d‘abord le tout puissant de m'avoir donné le courage et la patience jusqu'à l'achèvement de ce travail.

Je tiens à remercier plus particulièrement mon directeur de thèse Dr. Nadjib Badache, pour m’avoir fait confiance en me proposant ce sujet. Je le remercie également pour ses conseils et ses critiques.

Je tiens à remercier le président de jury Mr A. Aissani, ainsi que les membres de jury Mme Alimazighi, Mr Larabi et Mr Tandjaoui pour accepter d’évaluer mon travail.

J’adresse également mes sincères remerciements à Madame Hassina Aliane chef du laboratoire des logiciels de base au CERIST pour ses encouragements, sa compréhension et pour tous les moyens qu’elle a mis à ma disposition.

Mes remerciements s'adressent aussi à ma famille, mes amis et tous ceux qui m'ont aidé et contribué de prés ou de loin par une quelconque façon, que ce soit par leur amitié, leurs conseils ou leur soutient moral ou matériel.

Sommaire

Introduction générale

Les progrès technologiques récents ont permis aux utilisateurs de communiquer, et d'échanger des informations à travers des liaisons sans fil, ce qui leur permet ainsi de former un réseau mobile sans fil. Ces réseaux peuvent être classés en deux catégories : les réseaux avec infrastructure qui utilisent le modèle de communication cellulaire et qui exigent d'importantes infrastructures fixes, et les réseaux sans infrastructure appelés réseaux Ad Hoc.

Un réseau Ad Hoc est un réseau temporaire, formé d'un ensemble de machines qui peuvent communiquer entre elles sans aucune forme d'administration centrale. Il est important de noter que les réseaux ad hoc ne sont ni un remplacement ni une alternative des réseaux avec infrastructure. Ils complètent les réseaux avec infrastructure dans le cas où le coût d'établir ces réseaux serait plus élevé ou dans le cas où la durée d'installation ne justifierait pas de câblage à demeure.

Dans un réseau ad hoc, les mobiles sont connectés via des liens sans fil qui sont particulièrement vulnérables aux différentes attaques possibles. Cela se justifie par les contraintes et les limitations physiques (puissance de calcul et capacité de stockage limitées en plus des restrictions de la bande passante), qui font que le contrôle des données transférées doit être minimisé. En plus des menaces qui viennent du fait que les communications sans fil sont transmises par ondes radios et peuvent être écoutées par des personnes non autorisées.

Les problèmes de sécurité dans les réseaux ad hoc sont différents des problèmes dans les réseaux fixes, mais les besoins de sécurité (l'authentification, l'intégrité, la confidentialité, la disponibilité) sont les mêmes.

Dans ce travail, on s'intéresse au problème d'authentification dans les réseaux mobiles ad hoc. Autrement dit, on doit s'assurer que la personne à qui on parle est celle attendue. Ainsi, l'authentification est un problème très immédiat, et c'est là que résident les vraies questions d'aujourd'hui dans les réseaux ad hoc.

Ce document est composé de trois chapitres :

- Le premier chapitre est une introduction aux réseaux mobiles, et particulièrement aux réseaux mobiles ad hoc, on y trouve aussi une présentation du problème de routage dans ces réseaux.
- Le second est une description de quelques notions sur la sécurité réseau, et des techniques cryptographiques utilisées pour garantir les besoins de sécurité.
- Le troisième chapitre est une présentation de différentes solutions qui existent et qui résolvent le problème d'authentification dans les réseaux mobiles ad hoc.
- Dans le quatrième chapitre, on décrit les différentes étapes du protocole que nous avons proposé.

- Et enfin, dans le dernier chapitre, on présente une étude comparative de différents algorithmes cryptographiques, pour étudier la performance de notre protocole.

Chapitre 1

Les réseaux Ad Hoc et le problème de routage

Chapitre 1: Les réseaux mobiles ad hoc et le problème de routage

1.1 Introduction

L'informatique mobile a pris récemment un essor très important grâce aux avancées technologiques qui y sont liées.

Tout d'abord, les différents matériels portables, permettent désormais de travailler de manière de plus en plus indépendante, sans être contraint de conserver une liaison filaire pour assurer l'alimentation en électricité. Parallèlement, la miniaturisation et l'adaptation des composants permettent des gains en puissance de calcul et en autonomie (batteries de petite taille, stratégies d'économie d'énergie). Il existe actuellement un grand nombre de machines portables de types différents, allant du récepteur d'appel (pager) à la station de travail portable (ordinateur compatible).

Ensuite, en utilisant conjointement ces ordinateurs portables et les médiums de communications sans fil (ondes radio-fréquence, ondes lumineuses), il devient possible de rester connecté à son réseau habituel et de communiquer avec les autres machines mobiles tout en se déplaçant.

L'environnement de calcul résultant est appelé *environnement mobile*. Cet environnement n'astreint plus l'usager à une localisation fixe, mais il lui permet une libre mobilité tout en assurant sa connexion avec le réseau.

1.2 L'environnement mobile

Un environnement mobile est un système composé de sites mobiles et qui permet à ses utilisateurs d'accéder à l'information indépendamment de leurs positions géographiques **[BAD 98]**. Les réseaux mobiles ou sans fil, peuvent être classés en deux classes : les réseaux cellulaires avec infrastructure et les réseaux ad hoc sans infrastructure fixe.

Le modèle de *système avec infrastructure* est composé de deux ensembles d'entités distinctes :

- ✓ Les "**sites fixes**" du réseau filaire (wired network).
- ✓ Les "**sites mobiles**" (wireless network).

Certains sites fixes, appelés stations de base (SB) sont munis d'une interface de communication sans fil pour la communication directe avec les sites mobiles localisés dans une zone géographique limitée, appelée cellule (figure 1.1). A chaque station de base correspond une cellule à partir de laquelle des unités mobiles (UM) peuvent émettre et recevoir des messages. Alors que les sites fixes sont interconnectés entre eux à travers un réseau de communication filaire.

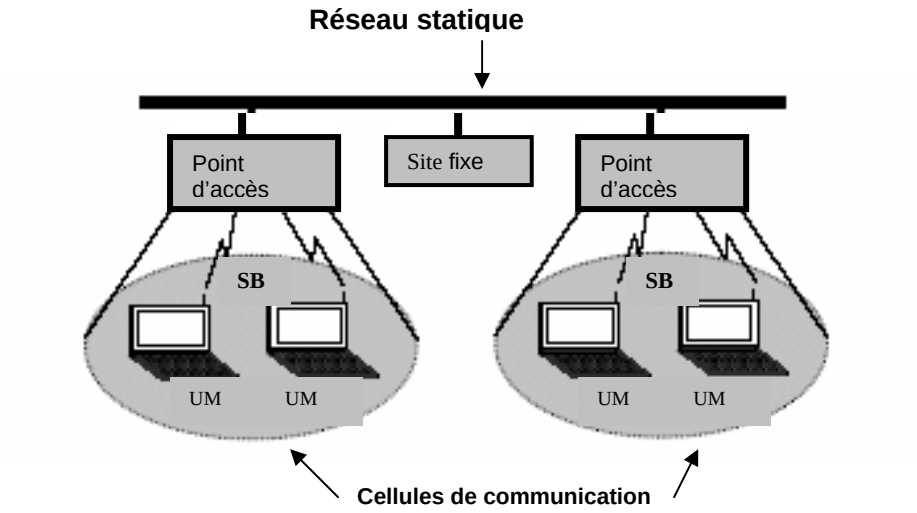

Fig. 1.1 : Les réseaux mobiles avec infrastructure

Une unité mobile ne peut être, à un instant donné, directement connectée qu'à une seule station de base. Elle peut communiquer avec les autres sites à travers la station à laquelle elle est directement rattachée **[BAD 98]**.

Dans le modèle de *réseau sans infrastructure* (figure1.2), l'entité "site fixe" n'existe pas, tous les sites du réseau sont mobiles et se communiquent d'une manière directe en utilisant leur interface de communication sans fil. L'absence de l'infrastructure ou du réseau filaire composé des stations de base, oblige les unités mobiles à se comporter comme des routeurs qui participent à la découverte et la maintenance des chemins pour les autres hôtes du réseau.

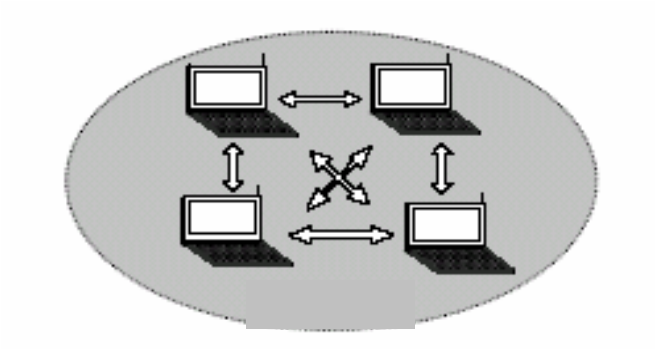

Fig. 1.2 : Un réseau mobile ad hoc sans infrastructure

1.3 Les Réseaux Mobiles Ad Hoc

L'évolution récente de la technologie dans le domaine de la communication sans fil et l'apparition des unités de calculs portables (les *laptops* par exemple), poussent aujourd'hui les chercheurs à faire des efforts afin de réaliser le but des réseaux : "l'accès à l'information n'importe où et n'importe quand" **[LEM 00]**.

Le concept des réseaux mobiles Ad Hoc essaie d'étendre les notions de la mobilité à toutes les composantes de l'environnement. Contrairement aux réseaux basés sur la communication cellulaire, aucune administration centralisée n'est disponible, les hôtes mobiles eux-mêmes forment, d'une manière Ad Hoc, une infrastructure du réseau. Aucune supposition ou limitation n'est faite sur la taille du réseau Ad Hoc, le réseau Ad Hoc peut contenir alors des centaines ou des milliers d'unités mobiles.

Un groupe d'unités portables reliées par des cartes HIPERLAN est un exemple d'un réseau ad hoc. Les réseaux appelés GSM ne représentent pas des réseaux ad hoc, car la communication entre les unités passe obligatoirement par des stations de base du réseau filaire **[HIP 02]**.

La mobilité des nœuds dans un réseau ad hoc, peut causer des changements fréquents dans la topologie du réseau. La topologie est donc dynamique et imprévisible, ce qui fait que la déconnexion des unités soit très fréquente. Par exemple dans la figure (1.3. a), les nœuds A et D ont un lien direct entre eux. Quand D quitte la portée de communication de A, le lien sera alors coupé, mais le réseau reste toujours en connexion, car A peut atteindre D par l'intermédiaire des nœuds C, E et F (figure 1.3. b).

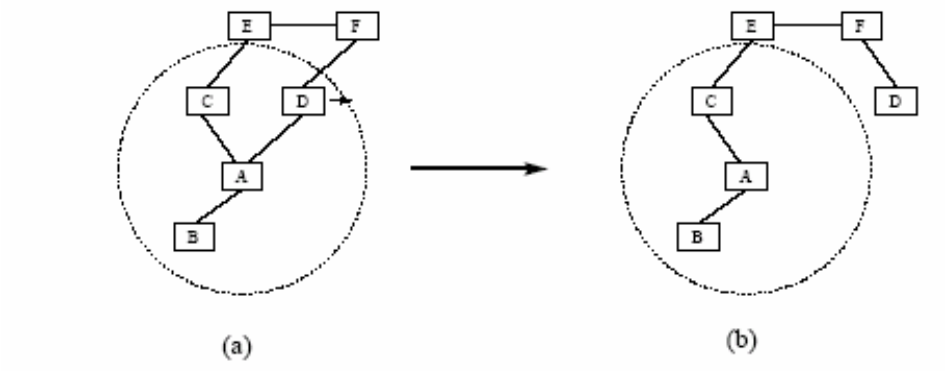

Fig. 1.3 : Un exemple de changement de topologie de réseau ad hoc

1.3.1 Définition

Un réseau mobile Ad Hoc, appelé généralement **MANET** (**M**obile **A**d Hoc **NET**work), consiste en une grande population, relativement dense, d'unités mobiles

qui se déplacent dans un territoire quelconque et dont le seul moyen de communication est l'utilisation des interfaces sans fil, sans l'aide d'une infrastructure préexistante ou administration centralisée.

Un réseau Ad Hoc peut être modélisé par un graphe **Gt = (Vt, Et)**. Où **Vt** représente l'ensemble des nœuds (les unités ou les hôtes mobiles) du réseau et **Et** modélise l'ensemble des connections qui existent entre ces nœuds. Si **e = (*u*, *v*)** $\in$ **Et**, alors les nœuds *u* et *v* sont en mesure de communiquer directement à un instant *t*. La figure 1.4 représente un réseau Ad Hoc de 9 unités mobiles sous forme d'un graphe :

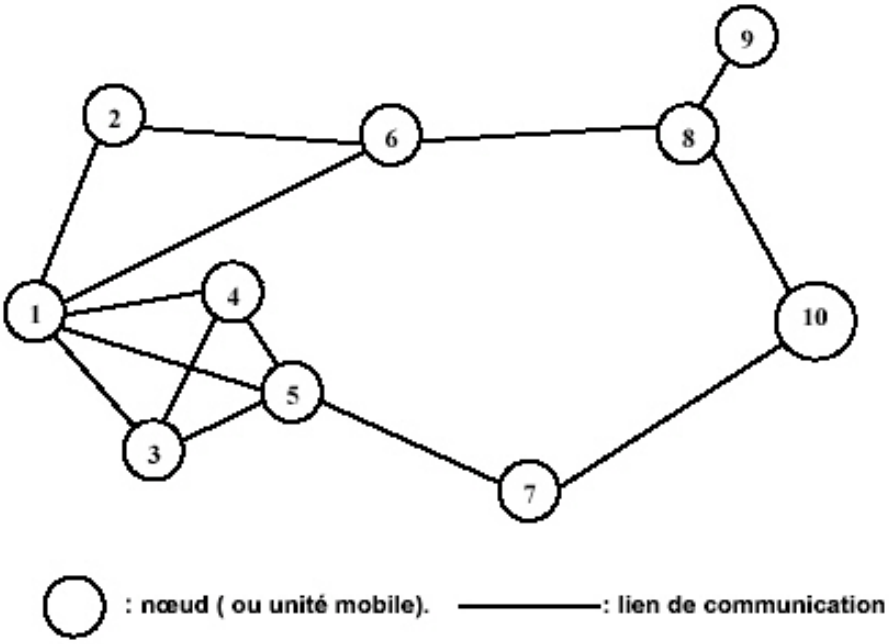

Fig. 1.4 : Modélisation d'un réseau ad hoc

1.3.2 Les domaines d'application des réseaux ad hoc

Les réseaux Ad Hoc sont idéals pour les applications caractérisées par une absence (ou la non fiabilité) d'une infrastructure préexistante, parmi ces applications, on cite :

• ***Les applications militaires*** **:** les équipements militaires (les soldats et les chefs de troupes) peuvent bénéficier des avantages de réseaux Ad Hoc, qui leur permettent de se déplacer et de communiquer entre eux dans les champs de bataille **[DER 01]**.

• ***Les opérations de secours*** (incendies, tremblement de terre…) et les missions d'exploration où une infrastructure du réseau filaire n'existe pas, mais une communication entre les équipes de secours est nécessaire. L'installation d'un réseau ad hoc dans ce cas peut résoudre le problème.

• ***Les conférences*** **:** un réseau ad hoc est utilisé par un groupe de gens qui veulent créer un réseau temporaire. Par exemple, un réseau qui relie les différentes unités

portables peut être utilisé pour partager les informations dans une conférence **[FRO 00]**.

• ***Contrôle d'environnement :*** des petits véhicules équipés de caméras et de détecteurs de son peuvent être utilisés dans une région déterminée afin de collecter un ensemble d'information et de l'envoyer à travers un réseau ad hoc à une station qui traite ces informations, par exemple on peut avec cette méthode prévoir la pollution de l'eau **[ROY 99]**.

D'une façon générale, les réseaux Ad Hoc sont utilisés dans toute application où le déploiement d'une infrastructure réseau filaire est trop contraignant, soit parce que c'est difficile de mettre en place un réseau filaire, soit parce que la durée d'installation de ce réseau ne justifie pas de câblage à demeure.

1.3.3 Caractéristiques des réseaux Ad hoc

Les réseaux mobiles Ad Hoc sont caractérisés par **[COR 99]** :

- ***Une 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.
- ***Une 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.
- ***Des 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.
- ***Une sécurité physique limitée*** **:** Les réseaux mobiles ad hoc sont plus vulnérables que les réseaux filaires classiques. Cela se justifie par les contraintes et les limitations physiques qui font que le contrôle des données transférées doit être minimisé.
- ***L'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.

1.4 Le problème de Routage dans les réseaux Ad Hoc

Lorsque le réseau ad hoc est géographiquement étendu, certaines stations peuvent être hors de portée les unes des autres. Ce cas nécessite l'emploi d'un routage interne des messages par les stations intermédiaires. La gestion de ce routage consiste à établir une architecture où l'on doit tenir compte de la mobilité des stations et de la versatilité du médium physique **[HIP 02]**.

1.4.1 Définition de routage

Le *routage* est une méthode d'acheminement des informations à la bonne destination à travers un réseau de connexion donné. Le problème de routage consiste à déterminer un acheminement optimal des paquets à travers le réseau en considérant certains critères de performance. Il s'agit d'établir des routes correctes et efficaces entre une paire quelconque d'unités, ce qui assure l'échange des messages d'une manière continue.

1.4.2 La difficulté du routage dans les réseaux Ad Hoc

Un réseau Ad Hoc étant caractérisé par une *absence d'infrastructure fixe* préexistante, à l'inverse des réseaux classiques, doit s'organiser automatiquement de façon à être déployable rapidement et pouvoir s'adapter aux conditions de propagation, au trafic et aux différents mouvements pouvant intervenir au sein des unités mobiles.

Dans le but d'assurer la connectivité du réseau, malgré l'absence d'infrastructure fixe et la mobilité des stations, chaque nœud doit être mis à contribution pour participer au routage et pour retransmettre les paquets d'un nœud qui n'est pas en mesure d'atteindre sa destination ; tout nœud joue ainsi le rôle de station et de routeur.

Chaque nœud participe donc à un *protocole de routage* qui lui permet de découvrir les chemins existants, afin d'atteindre les autres nœuds du réseau. Le fait que la taille d'un réseau Ad Hoc peut être énorme, souligne que la gestion de routage de l'environnement doit être complètement différente des approches utilisées dans le routage classique. Le problème qui se pose dans le contexte des réseaux ad hoc est l'adaptation de la méthode d'acheminement utilisée avec le grand nombre d'unités existant dans un environnement caractérisé par de modestes capacités de calcul et de sauvegarde.

Dans la pratique, il est impossible qu'un hôte puisse garder les informations de routage concernant tous les autres nœuds, dans le cas où le réseau serait volumineux. Certains protocoles de routage utilisent la sauvegarde des données de routage concernant une destination donnée. Cependant, ces protocoles ne spécifient pas les destinations (leurs données de routage) que les nœuds doivent garder. Le problème ne se pose pas dans le cas de réseaux de petites tailles, car l'inondation (la diffusion pure) faite dans ces réseaux n'est pas coûteuse. Par contre dans un réseau volumineux, le manque de données de routage concernant les destinations peut impliquer une diffusion énorme dans le réseau, et cela si on considère seulement la phase de découverte de routes. Le trafic causé par la diffusion, dans ce cas, est rajouté au trafic déjà existant dans le réseau, ce qui peut dégrader considérablement les performances de transmission du système caractérisé principalement par une faible bande passante.

Dans le cas où le nœud destination se trouverait dans la portée de communication du nœud source, le routage devient évident et aucun protocole de routage n'est initié. Malheureusement, ce cas est généralement rare dans les

réseaux Ad Hoc Une station source peut avoir besoin de transférer des données à une autre station qui ne se trouve pas dans sa portée de communication.

Par exemple, dans la figure 1.5, l'unité mobile W n'est pas dans la portée de communication de l'unité U (indiquée par le cercle d'origine U) et vice versa. Dans le cas où l'unité U veut transférer des paquets à W, elle doit utiliser les services de l'unité V dans l'envoi des paquets, puisque l'unité V contient dans sa portée de communication les unités U et W.

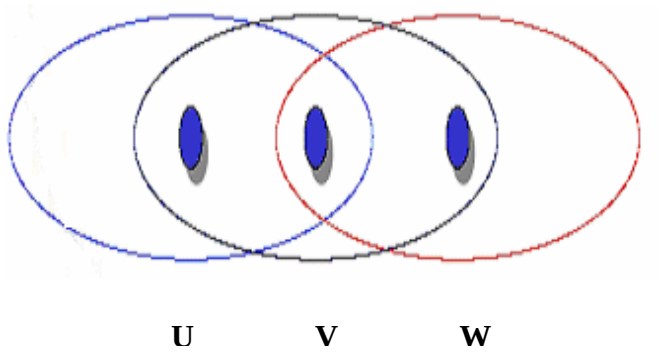

Fig. 1.5 : Un simple réseau Ad Hoc constitué de trois unités mobiles.

Dans la pratique, le problème de routage est plus compliqué à cause de la non uniformité de la transmission sans fil et de la possibilité du déplacement imprévisible de tous les nœuds concernés par le routage.

1.4.3 La conception de stratégie de routage

Le problème qui se pose dans les réseaux Ad Hoc est l'adaptation de la méthode d'acheminement utilisée avec le grand nombre d'unités existant dans un environnement. Assurer la connexion des réseaux ad hoc au sens classique du terme (tout sommet peut atteindre tout autre), est un problème complexe : *l'environnement est dynamique et la topologie du réseau peut changer fréquemment.* Par conséquent, il est important que toute conception de protocole de routage étudie les problèmes suivants :

La minimisation de la charge du réseau : l'optimisation des ressources du réseau renferme deux autres sous problèmes qui sont l'évitement des boucles de routage, et l'empêchement de la concentration du trafic autour de certains nœuds ou liens.

Offrir un support pour pouvoir effectuer des communications multipoints fiables: le fait que les chemins utilisés pour router les paquets de données puissent évoluer, ne doit pas avoir d'incident sur le bon acheminement des données. L'élimination d'un lien, pour cause de panne ou pour cause de mobilité devrait, idéalement, augmenter le moins possible les temps de latence.

Assurer un routage optimal : la stratégie de routage doit créer des chemins optimaux et pouvoir prendre en compte différentes métriques de coûts (bande passante, nombre de liens, ressources du réseau, délais de bout en bout, etc.). Si la construction des chemins optimaux est un problème dur, la maintenance de tels chemins peut devenir encore plus complexe, la stratégie de routage doit assurer une maintenance efficace de routes avec le moindre coût possible.

Le temps de latence : la qualité des temps de latence et de chemins doit augmenter dans le cas où la connectivité du réseau augmenterait.

1.5 Classification des protocoles de routage pour les réseaux Ad Hoc

Les protocoles de routage peuvent être séparés en deux catégories, suivant la manière de création et de maintenance de routes lors de l'acheminement des données **[MAL 01]**, voir figure 1.6 :

- *Les protocoles proactifs (Table Driven) :* établissent les routes à l'avance en se basant sur l'échange périodique des tables de routage.

- *Les protocoles réactifs (On Demand)* : cherchent les routes à la demande.

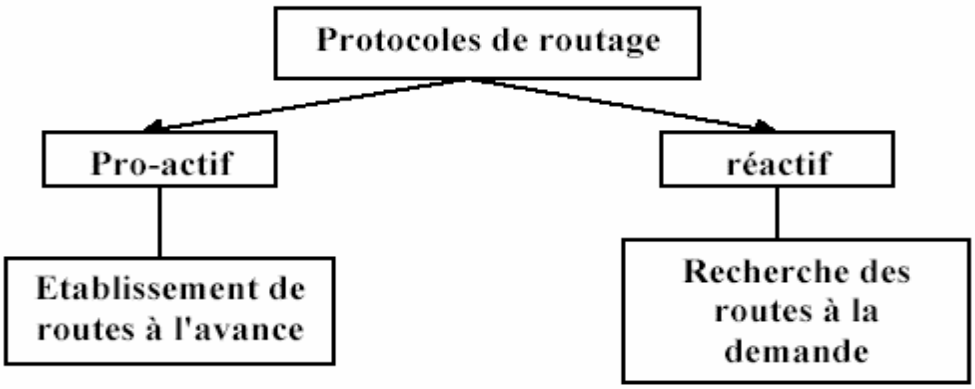

Fig. 1.6 : Classification des protocoles de routage.

1.5.1 Les protocoles de routage proactifs

Dans les réseaux mobiles ad hoc, les protocoles de routage proactifs sont basés sur le même principe des protocoles de routage utilisés dans les réseaux filaires conventionnels. Pour cela, nous allons présenter les deux méthodes principales utilisées dans le routage des réseaux filaires : la méthode *Etat de Lien* ("Linka State") et la méthode du *Vecteur de Distance* ("Distance Vector"). Ces deux méthodes exigent une mise à jour périodique des données de routage **[DER 01, ROY 99]**.

Les algorithmes de routage basés sur ces deux méthodes, utilisent la même technique qui est la technique des plus courts chemins, et permettent à un hôte donné, de trouver le prochain hôte pour atteindre la destination en utilisant le trajet le plus court existant dans le réseau.

Généralement le calcul du plus court chemin entre deux stations, est basé sur le nombre de nœuds (ou de sauts) que comportent les différents chemins qui existent entre les deux stations. On peut aussi associer des coûts aux liens de communication. Un coût peut matérialiser le taux de l'utilisation d'un lien, les délais relatifs de communication ou un autre facteur qu'on veut minimiser lors du routage des données.

Les protocoles de routage proactifs essaient de maintenir les meilleurs chemins existants vers toutes les destinations possibles au niveau de chaque nœud du réseau. Ainsi, les chemins de routage sont sauvegardés même s'ils ne sont pas utilisés. La sauvegarde permanente des chemins de routage, est assurée par un échange continu des messages de mise à jour des chemins, ce qui induit un contrôle excessif surtout dans le cas des réseaux de grande taille **[DER 01]**.

1.5.2 Les protocoles de routage réactifs

Les protocoles de routage appartenant à cette catégorie permettent de créer et de maintenir les routes selon les besoins. Lorsque le réseau a besoin d'une route, une procédure de découverte globale de routes est lancée, et cela dans le but d'obtenir une information spécifiée, inconnue au préalable.

Les protocoles de routage réactifs appelés aussi protocoles de routage à la demande, représentent les protocoles les plus récents, proposés dans le but d'assurer le service du routage dans les réseaux sans fil.

Les deux techniques de routage proactif et réactif, ont leurs avantages et inconvénients, selon le type d'application et l'environnement d'implémentation. Le routage à la demande induit une lenteur, à cause de la recherche des chemins, ce qui peut dégrader les performances des applications interactives (par exemple, les applications de bases de données distribuées), mais ces protocoles sont plus adaptatifs que les protocoles proactifs **[DER 01]**.

Les protocoles des deux catégories essaient de s'adapter aux contraintes imposées par le réseau ad hoc, et cela en proposant une méthode qui soit de moindre coût en capacités et ressources. Les protocoles de routage offrent différents avantages qui sont en réalité complémentaires et préférables pour différents types d'applications **[LEM 00, ROY99]**.

1.6 Conclusion

Dans ce chapitre nous avons présenté le concept des réseaux ad hoc et le problème de routage dans cet environnement. Dans la pratique, les réseaux ad hoc connaissent aujourd'hui plusieurs applications telles que les applications militaires, les applications de secours et de façon générale, toutes les applications caractérisées par une absence d'infrastructure préexistante.

Après avoir défini l'environnement mobile ad hoc et décrit ses principales applications et caractéristiques, nous avons présenté le problème d'acheminement des paquets dans les réseaux ad hoc, c'est à dire le problème de routage. Le routage est un service très important dans les environnements mobiles, puisqu'il n'y a pas une infrastructure qui s'occupe de l'acheminement de données.

L'étude des réseaux mobiles, et spécialement des réseaux mobiles ad hoc montre qu'un environnement mobile est très différent d'un environnement filaire traditionnel. Dans beaucoup de cas, les mobiles sont connectés à un réseau via des liens sans fil, qui sont particulièrement vulnérables aux différentes attaques possibles. Cela se justifie par les contraintes et limitations physiques (puissance de calcul et capacité de stockage limitées, en plus des restrictions de la bande passante), qui font que le contrôle des données transférées doit être minimisé. En plus des menaces qui viennent du fait que les communications sans fil sont transmises par ondes radios et peuvent être écoutées par des personnes non autorisées.

La mise en œuvre de politiques et de procédures de sécurité efficaces est la première étape pour contrôler les risques que présentent les systèmes sans fil. Ainsi, nous allons introduire dans le chapitre suivant, quelques notions de sécurité, ensuite nous allons définir les techniques et les outils cryptographiques, qui permettent de garantir la sécurité.

Chapitre 2

la sécurité réseau : notions et techniques

Chapitre 2 : La sécurité réseau : notions et techniques

2.1 Introduction

Le succès de plus en plus important des réseaux sans fil, nous amène naturellement à s'intéresser au problème de leur sécurité. Les réseaux mobiles sans fil sont généralement plus sensibles aux menaces physiques que les réseaux fixes câblés. Les techniques de sécurité existantes au niveau des liaisons filaires (comme par exemple, le chiffrement) sont souvent appliquées au sein des réseaux sans fil pour réduire les menaces d'attaques.

Ce chapitre est une initiation à la sécurité réseau. Il introduit la notion de cryptographie (chiffrement) et présente quelques techniques et outils de la sécurité. Finalement, quelques protocoles de sécurité sont décrits.

2.2 Les services de la sécurité

Pour mettre en œuvre un système sécurisé, il faut assurer les services de sécurité suivants **[BID 95]** :

- ***La confidentialité :*** assure qu'une information transmise n'est accédée que par une personne autorisée.
- ***L'intégrité :*** assure que les données transmises ne sont pas altérées au cours de la transmission.
- ***L'authentification :*** permet de s'assurer de l'identité du correspondant, c'est à dire vérifier qu'il est bien celui qu'il dit être.
- ***La disponibilité :*** est la possibilité d'accéder à des ressources autorisées.
- ***Non répudiation* :** l'expéditeur d'un message ne peut pas nier l'émission.

2.3 Les types d'attaques

Les attaques sur la sécurité peuvent être classées en deux catégories selon la nature de l'attaquant :

- *Les attaques passives :* l'attaquant peut seulement écouter ou surveiller le trafic des réseaux. C'est la forme la plus facile des attaques et peut être exécutée sans difficulté dans beaucoup d'environnements de réseau, par exemple le réseau Ethernet et les réseaux sans fil.

- *Attaques actives :* l'attaquant peut non seulement écouter la transmission mais peut également la modifier ou la supprimer.

De plus, selon les activités d'un attaquant, les sous catégories suivantes peuvent être utilisées pour couvrir la majorité d'attaques possibles **[FOK 02]** :

Écoute passive (message eavesdropping) : est un type d'attaque contre la confidentialité, qui consiste à accéder sans modification aux informations transmises

ou stockées ; l'information n'est pas altérée par celui qui en prélève une copie. Ces attaques sont donc indétectables par le système et peuvent seulement être parées par des mesures préventives. La solution est de rendre inintelligible les messages transmis.

Analyse de trafic (Traffic analysis) : Le but principal de cette attaque ne consiste pas à accéder aux données transmises, mais d'extraire des informations utiles à partir des caractéristiques de la transmission, par exemple la quantité des données transmises et l'identité des nœuds communicants. Ces informations peuvent permettre à l'attaquant de déduire l'information sensible, par exemple les rôles des nœuds communicants, leurs positions. À la différence de l'attaque précédente, il est plus difficile d'empêcher cette attaque.

Interposition : L'attaquant utilise l'identité d'un autre nœud pour accéder à une ressource ou à des données non autorisées. Cette attaque consiste à tromper les mécanismes d'authentification pour se faire passer pour un utilisateur légitime.

Modification : cette attaque modifie des données transmises entre les nœuds communicants. Un exemple pourrait être une transaction financière, où l'attaquant modifie une valeur de la transaction.

Insertion (insertion) : Cette attaque implique une partie non autorisée, qui insère de nouvelles données réclamant qu'elle est de la part d'une partie légitime.

Rejoue (Replay) : L'attaquant retransmet des données précédemment envoyées par un nœud légitime.

Déni de service : Cette attaque active vise à limiter l'accès à une certaine ressource autorisée. Cette ressource peut être un nœud spécifique, un service ou le réseau entier.

2.4 Les outils cryptographiques

Les services de sécurité décrits précédemment, peuvent être assurées en utilisant différentes techniques cryptographiques :

2.4.1 Le cryptage symétrique (ou à clé secrète)

On parle de cryptage symétrique, lorsqu'on utilise une même clé pour crypter et décrypter les messages (figure 2.1). Le principal inconvénient de ce système est la difficulté de la distribution de la clé de manière sécurisée. Car il suffit qu'une personne puisse intercepter cette clé durant son envoi au destinataire pour qu'elle puisse décrypter tous les messages cryptés avec cette clé **[ZIM 95]**.

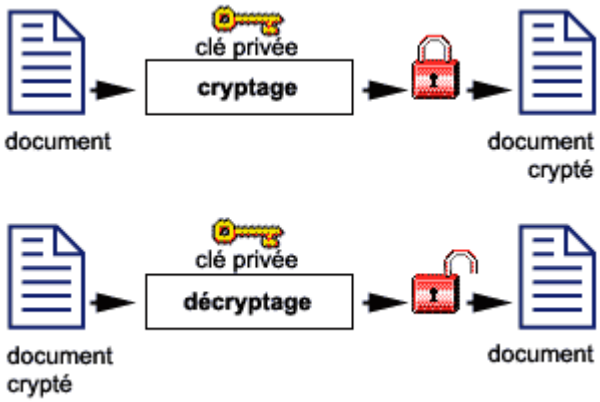

Fig. 2.1 : Le cryptage symétrique

2.4.2 Le cryptage asymétrique (ou à clé publique)

Le cryptage asymétrique résout le problème de partage de clés, car chaque utilisateur comporte deux clés liées mathématiquement (figure 2.2) :

- Une clé privée qui doit être gardée secrète.
- Une clé publique qui est disponible pour tous les autres utilisateurs, utilisée pour le cryptage de données.

Fig. 2.2 : Le cryptage asymétrique

Dans la pratique, la clé publique sert à crypter les messages, et la clé privée sert à les décrypter. Une fois le message crypté, seul le destinataire est en mesure de le décrypter.

Supposons par exemple que B veuille envoyer un message confidentiel à A.

- B crypte les données pour A avec la clé publique de A.
- A est donc le seul à comprendre les données car il détient sa clé privée.

Le cryptage à clé publique présente deux inconvénients **[KLE 00]** :

- Il est plus lent que le cryptage à clé secrète.
- Il nécessite une grande puissance de calcul.

2.4.3 Les fonctions de hachage

Une fonction de hachage est une fonction permettant d'obtenir un condensat à partir d'un message, c'est-à-dire une suite de caractères assez courte représentant le message à envoyer. La fonction de hachage doit être telle qu'elle associe un et un seul condensat à un message en clair (cela signifie que la moindre modification du message entraîne la modification de son condensat).

En expédiant un message accompagné de son condensat, il est possible de garantir l'intégrité de ce message, c'est-à-dire que le destinataire peut vérifier que le message n'a pas été altéré (intentionnellement ou accidentellement) durant la communication.

2.4.4 La signature numérique

La façon la plus simple pour signer un message consiste à chiffrer celui-ci à l'aide d'une clé privée : seul le possesseur de cette clé est capable de générer la signature, mais toute personne ayant accès à la clé publique correspondante peut la vérifier. Dans la pratique, cette méthode est peu utilisée du fait de sa lenteur [BID 95].

Pour cette raison, une autre méthode est utilisée pour signer, consiste à calculer un condensat du message à signer et à ne chiffrer que ce condensat. Le résultat obtenu s'appelle la signature numérique. Le destinataire reçoit le message et la signature. Il décrypte d'abord la signature à l'aide d'une clé publique fournie par l'expéditeur au préalable. Il obtient ainsi un condensat. A partir du message, il exécute le même algorithme de hachage que l'expéditeur pour obtenir un condensat (figure 2.3). Si les deux condensats ainsi obtenus sont identiques, l'expéditeur aura la certitude que le message est authentique (sous la condition que la clé publique utilisée soit bien celle envoyée par l'expéditeur).

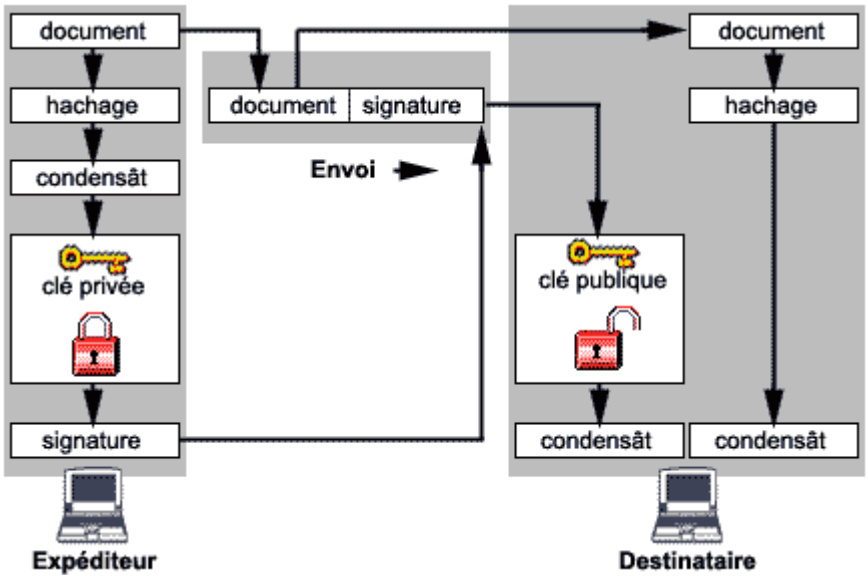

Fig. 2.3 : Mécanisme de signature numérique

On peut dire donc que cette signature permet de vérifier l'identité de l'expéditeur du message qui est le seul à pouvoir créer la signature grâce à sa clé privée.

2.4.5 Les certificats numériques

L'un des plus grands problèmes de la cryptographie asymétrique est l'authentification des clés publiques. Car n'importe qui peut envoyer une clé en prétendant qu'elle appartient à une autre personne dans le but d'intercepter et de déchiffrer tous les messages envoyés à cette personne **[ZIM 95]**. Dans un environnement à clé publique, il est vital de s'assurer que les clés utilisées ne sont pas des contrefaçons. Les certificats numériques simplifient la tâche d'établir la réelle appartenance d'une clé à son propriétaire supposé **[BER 00]**. Un certificat numérique est une information attachée à une clé publique. Il comporte les éléments suivants :

- Une clé publique.
- Les informations d'identification (l'identité de l'utilisateur, son mail, etc.)

Ces informations sont complétées par le format et le numéro de série de certificat, sa période de validité ainsi que par le type d'algorithme associé à la clé publique de l'utilisateur. Dès lors est calculée une valeur numérique (via une fonction de hachage) qui est un condensat unique des données (figure 2.4). Ce condensat est ensuite chiffré par le biais de la clé privée de l'autorité de certification.

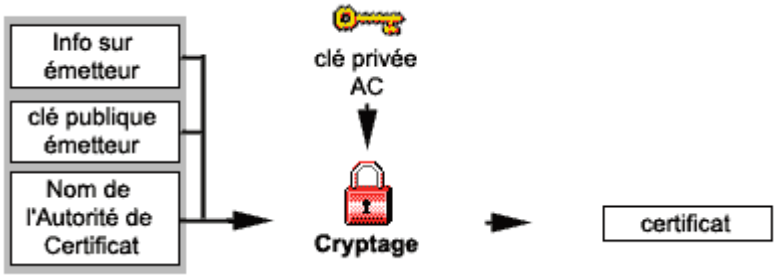

Fig. 2.4 : Création de certificat numérique

2.4.6 La notion de partage de secret

Une autre application de la cryptographie, appelée "*partage de secret*", consiste à partager un secret entre un certain nombre d'individus appartenant à un groupe. Le but consiste à rendre nécessaire la présence simultanée d'un nombre minimum fixé d'individus appartenant à ce groupe pour rendre le secret accessible. Par exemple, dans un système à seuil «threshold system» (k, n), l'information sur un secret est distribuée de telle façon que n'importe quel ensemble de k personnes parmi n ($k < n$) a assez d'informations pour déterminer le secret, mais n'importe quel ensemble de $(k-1)$ personnes ne le peut pas. Dans tout procédé de partage de secret, il y a des

ensembles donnés de personnes dont l'information cumulative suffit à déterminer le secret. Dans certaines mises en oeuvre de procédés de partage de secret, chaque participant reçoit le secret une fois qu'il a été généré. Dans d'autres réalisations, le secret véritable n'est jamais vu par les participants **[SCH 96]**.

2.5 Mécanismes de sécurité

Parmi les mécanismes de sécurité existants, on décrit dans ce qui suit les deux systèmes PKI (Public Key Infrastructure) et PGP (Pretty Good Privacy), on présente aussi l'algorithme d'échange de clés Diffie Hellman.

2.5.1 Algorithme d'échange de clé (Diffie Hellman)

L'algorithme Diffie-Hellman est utilisé pour la distribution de clés. Par exemple, deux entités Alice et Bob peuvent utiliser cet algorithme pour engendrer une clé secrète. Au départ Alice et Bob se mettent d'accord sur deux grands entiers, p et g, de telle manière que g soit inférieur à p mais plus grand que 1. Ces deux entiers n'ont pas à être secrets ; Alice et Bob peuvent convenir de ses nombres sur un canal non sûr.

Le protocole illustré dans la figure 2. 5, se déroule ainsi **[SCH 96] :**

1. Alice choisit un grand nombre entier aléatoire x_{Alice} et calcule :

$$Y_{Alice} = g^{X}_{Alice} \bmod p.$$

2. Bob choisit un grand nombre entier aléatoire x_{Bob} et calcule :

$$Y_{Bob} = g^{X}_{Bob} \bmod p.$$

3. Alice envoie Y_{Alice} à Bob et Bob envoie Y_{Bob} à Alice. On remarque que Alice garde x_{Alice} secret et Bob garde x_{Bob} secret.

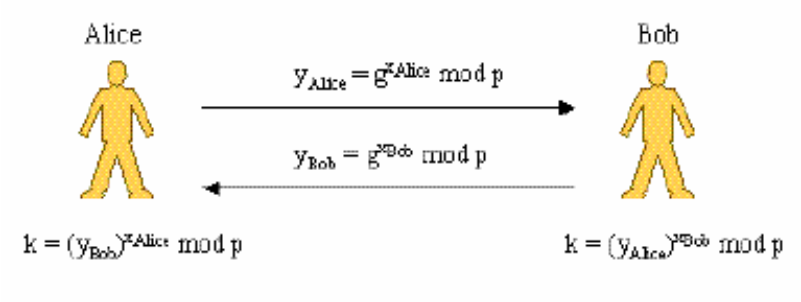

Fig. 2.5 : Echange de clé par l'algorithme Diffie-Hellman

4. Alice calcule $k = Y_{Bob}{}^{xAlice} \bmod p$.

Bob calcule k` = Y_{Alice}^{xBob} mod p.

Les valeurs k et k` sont toutes les deux égales à $g^{xAlice\ xBob}$ mod p. Personne ne peut en écoutant la communication calculer cette valeur ; celui qui écoute ne connaît que p, g, Y_{Alice} et Y_{Bob}. A moins qu'il ne puisse calculer le logarithme discret et retrouver x_{Alice} ou x_{Bob}. Ainsi k est la clé secrète que Alice et Bob ont calculée indépendamment.

2.5.2 Infrastructure à clé publique (PKI)

Une PKI (Public Key Infrastructure) est une infrastructure de gestion de certificats et de clés. Elle est composée d'un ensemble d'entités et d'utilisateurs, qui doivent communiquer entre eux en toute sécurité.

Trois types d'entités sont définis dans une PKI (voir figure 2.6) :

- ***Les utilisateurs ou entités finales :*** Ces utilisateurs ne représentent pas seulement les personnes possédant un certificat, mais aussi les applications qui utilisent ces certificats. Ces entités finales doivent disposer d'un moyen de stockage sûr pour sauvegarder leur clé privée.
- ***L'autorité de certification*** (Certification Authority ou CA) : est un organisme qui gère les certificats des différents utilisateurs.
- ***L'autorité d'enregistrement*** (Registration Authority ou RA) : est un organisme qui génère les demandes de certification d'un utilisateur. L'enregistrement de cet utilisateur n'est validé qu'après vérification des informations concernant cet utilisateur.

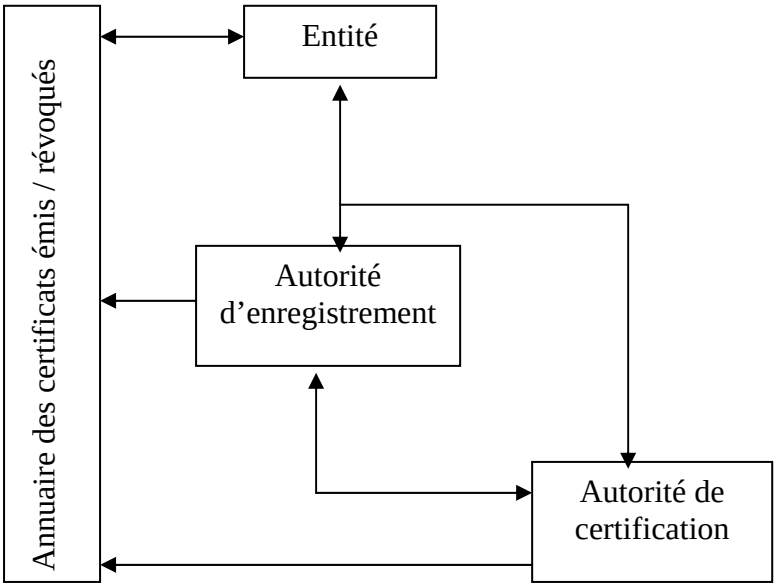

Fig. 2.6 : Les composants essentiels d'une PKI

Une PKI doit fournir les services suivants **[ARC 00]**:

Enregistrement : l'autorité d'enregistrement *vérifie l'identité du demandeur* de certificat ; elle s'assure que celui-ci possède bien un couple de clés privée / publique et récupère la clé publique du demandeur. Elle *transmet ensuite ces informations* (informations d'identité du demandeur ainsi que sa clé publique) à *l'autorité de certification*.

Initialisation : Pour qu'une entité puisse utiliser les services fournis par une PKI, elle doit être initialisée avec certaines informations. L'information la plus importante est le certificat de l'autorité de certification qui contient la clé publique, nécessaire à la vérification des certificats générés par cette autorité. L'initialisation implique aussi la génération des paires de clés publique / privée.

Certification : l'autorité de certification reçoit les demandes de création de certificats venant des autorités d'enregistrement. *Elle crée les certificats et elle les signe en utilisant sa clé privée*. Elle envoie ces certificats aux utilisateurs et en parallèle les transmet au service de publication. Une autorité de certification a donc un couple de clé privée/publique pour signer les certificats.

Renouvellement : Quand un certificat arrive à la date d'expiration[1], une entité doit pouvoir demander un nouveau certificat et une nouvelle paire de clé.

Révocation : l'autorité de certification est responsable de la maintenance des certificats qu'elle a délivrés. Par exemple, si une clé privée est révoquée et par conséquent le certificat devient invalide, l'autorité de certification doit le révoquer.

Publication : ce service rend disponible les certificats émis par l'autorité de certification. Il publie alors la liste des certificats valides et des certificats révoqués (qui ne sont plus valide) dans un annuaire de certificats, voir figure (2.6).

2.5.3 PGP (Pretty Good Privacy)

PGP est un système cryptographique hybride, qui combine à la fois le cryptage symétrique et le cryptage asymétrique. Il a été créé par Phil Zimmermann dans le but de préserver la sécurité des communications.

Le système PGP, à la différence des autres systèmes cryptographiques, peut offrir aux utilisateurs le droit de signer et de valider[2] les certificats des autres utilisateurs. Quand un utilisateur est sûr qu'un certificat est valide, il signe une copie du certificat pour attester le fait qu'il l'a contrôlé et qu'il est valide. S'il désire que d'autres personnes sachent qu'il a donné à ce certificat sa propre approbation, il envoie sa signature vers un serveur de certificats, pour que d'autres puissent l'utiliser **[ZIM 95]**.

[1] Lorsque l'autorité de certification délivre un certificat, celui-ci contient une date de création et une date de fin de validité.

[2] Ici, la validité d'un certificat est la croyance qu'une clé publique appartient réellement à son propriétaire présumé.

Il est possible de s'assurer manuellement de la validité d'un certificat en vérifiant son empreinte digitale. Chaque certificat PGP a une empreinte digitale unique déterminée par une fonction de hachage tout comme pour les signatures digitales.

Une autre façon d'établir la validité du certificat d'une clé, est de faire confiance à un troisième utilisateur qui a lui-même validé le certificat. Un utilisateur peut faire confiance à des gens qui valident des certificats ou qui font eux-mêmes aussi confiance à d'autres personnes qui valident des certificats. Une véritable toile de confiance peut donc se créer et la validation manuelle des certificats peut être évitée dans plusieurs cas.

A l'aide de PGP, un utilisateur peut choisir de faire confiance à certains utilisateurs. Le niveau de confiance accordé à ces derniers est personnel et est déterminé individuellement. Par exemple, un utilisateur pourrait accorder une totale confiance en son patron mais n'accorder qu'une confiance partielle en son voisin. Il fait alors confiance aux certificats validés par son patron, mais il doute des certificats validés par son voisin.

2.6 Conclusion

La sécurité a été toujours un facteur important dans la conception des réseaux, et elle est encore plus quand le réseau est sans fil (l’environnement des terminaux sans fil). La sécurité inclut aussi bien la sécurité des données contre des pertes et des corruptions, que leur confidentialité. Il faut donc protéger les données dans leurs points de stockages, ainsi que durant les transmissions. Les techniques classiques tel que le cryptage peuvent être utilisées, mais à cause des contraintes de l’environnement, les solutions de sécurité sont loin d’être évidentes.

Dans ce chapitre, nous avons présenté quelques protocoles de sécurité qui permettent de garantir les besoins de sécurité dans les réseaux filaires.

Dans le chapitre suivant, nous allons décrire quelques travaux de recherches récents, qui présentent des solutions permettant de garantir l’authentification et de gérer les clés dans les réseaux ad hoc.

Chapitre 3

L'authentification dans les réseaux Ad Hoc

Chapitre 3 : L'authentification dans les réseaux ad hoc

3.1 Introduction

L'architecture d'un réseau mobile ad hoc est caractérisée par une absence d'infrastructure fixe préexistante, à l'inverse des réseaux filaires classiques. Un réseau ad hoc peut s'organiser automatiquement de façon à être déployable rapidement et avec un coût relativement faible. Par conséquent, un réseau ad hoc est devenu intéressant par différentes applications (par exemple, les applications commerciales), autres que les applications militaires et les applications de secours.

Cependant, les propriétés importantes qui caractérisent un réseau ad hoc, tel que le manque d'infrastructure, représentent des inconvénients pour garantir la sécurité de ce type de réseau.

Les mécanismes de sécurité traditionnels, comme la signature digitale et le chiffrement à clé publique, reste toujours des outils essentiels pour garantir les besoins de sécurité dans les réseaux mobiles ad hoc. Ces mécanismes nécessitent un service de gestion de clés afin de garder une liaison entre une clé et un nœud (authentifier les clés utilisées), et d'établir une confiance entre les nœuds du réseau.

Traditionnellement, le service de gestion de clé était basé sur une entité digne de confiance, appelé autorité de certification CA qui doit créer un certificat de clé publique à chacun des nœuds. La CA digne de confiance doit être en ligne pour traiter les cas de révocation et de renouvellement des certificats de clés publiques. Cependant il est dangereux d'installer un service de gestion de clés en utilisant une seule CA dans un réseau ad hoc. Car si cette unique autorité de certification CA est compromise, la sécurité de tout le réseau est brisée. Le problème donc est comment installer un service de gestion de clés dans un réseau ad hoc **[ZHO 99, YAN 01]**.

Pour résoudre ce problème, la plus part des travaux actuels proposent de partager un secret parmi les nœuds qui forment le réseau. Le partage de secret diffère d'une solution à une autre.

Différentes solutions sont proposées :

3.1 Autorité de certification partiellement distribuée

3.1.1 Introduction

Dans l'article **[ZHO 99]**, les auteurs ont suggéré d'utiliser une infrastructure à clé publique distribuée pour gérer les clés utilisées dans le réseau, ainsi pour garantir l'authentification des nœuds qui forment ce réseau. Les schémas à clé secrète sont utilisés ultérieurement pour sécuriser les communications, après que les nœuds ne seraient authentifiés **[CAP 03]**.

Dans une infrastructure à clé publique, chaque nœud a une paire de clés publique/privée. Les clés publiques peuvent être distribuées aux autres nœuds tandis que les clés privées doivent être gardées secrètes. Ainsi, une autorité de certification digne de confiance (Certification Authority CA) est utilisée pour la gestion de clés et la création des certificats associés aux clés publiques.

En effet, l'Autorité de Certification (AC) doit rester *toujours en ligne*, pour renouveler les certificats des utilisateurs. Le renouvellement des certificats est nécessaire dans le cas où la clé publique serait compromise ; donc elle doit être révoquée ou dans le cas où le possesseur de cette clé quitte le réseau. D'autre part, un nœud doit renouveler sa paire de clés périodiquement pour réduire les chances d'attaques sur sa clé privée.

L'utilisation d'une seule autorité de certification dans un réseau ad hoc, pour établir un service de gestion de clés, peut causer des problèmes, car l'autorité de certification qui est responsable de la sécurité sera donc le point vulnérable du réseau **[ZHO 99]**. Si l'autorité de certification est compromise, sa clé privée est connue par un attaquant, ce dernier peut alors signer des certificats erronés en utilisant la clé privée attaquée, et peut aussi révoquer un certificat valide. D'autre part, dans le cas où l'autorité de certification ne serait plus disponible, les nœuds ne peuvent pas avoir les clés publiques des autres nœuds, donc ils ne peuvent pas établir une communication sécurisée avec les autres.

Pour résoudre ce problème, les auteurs de cet article **[ZHO 99],** proposent une réplication de l'autorité de certification (CA) dans le but d'améliorer la disponibilité de l'autorité de certification CA, mais une réplication naïve peut rendre le service plus vulnérable ; car il suffit de compromettre une seule réplication qui possède la clé privée du service, pour mener à l'effondrement de tout le système. Pour résoudre ce problème, les auteurs de cet article, ont proposé de distribuer un secret ou une confiance (trust) à un ensemble de nœuds, de manière à partager la responsabilité de gestion de clés entre les nœuds choisis. Avec cette proposition, la disponibilité de l'AC sera améliorée et par conséquent, l'AC ne sera pas le seul point vulnérable à attaquer.

3.1.2 Description du système

Le service de gestion de clés sera donc constitué de *n* nœuds choisis parmi les nœuds du réseau, appelés serveurs (servers), voir figure (3.1). Ce service de gestion de clés a une paire de clés publique / privée (*K* / *k*). La clé publique *K* est connue par tous les nœuds qui constituent le réseau, tandis que la clé privée *k* est divisée en *n* parts s_1, s_2, …, s_n, une part pour chaque serveur. Chaque serveur *i* possède aussi une paire de clés publique / privée K_i / k_i et connaît les clés publiques des autres nœuds **[ZHO 99]**, et en particulier les clés publiques des autres serveurs. Ainsi, les serveurs peuvent établir des liens sécurisés entre eux.

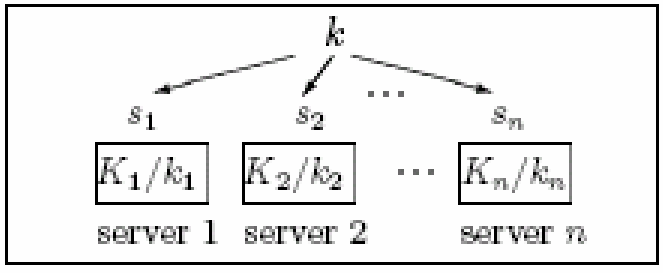

Fig. 3.1 : Service de gestion de clés *K* / *k*

Le système proposé dans cet article est composé de trois types de nœuds ; les nœuds clients, les nœuds serveurs et les combinateurs (figure 3.2). Les nœuds clients sont les utilisateurs du réseau ad hoc, tandis que les nœuds serveurs et les combinateurs sont des composants de l'autorité de certification. Les nœuds serveurs permettent de générer les certificats partiels et de les stocker, permettant ainsi aux nœuds clients d'acquérir les certificats des autres nœuds[1]. Les nœuds combinateurs qui peuvent aussi être des nœuds serveurs, sont responsables de combiner les certificats partiels pour former un certificat valide **[ZHO 99**, **FOK 02]**.

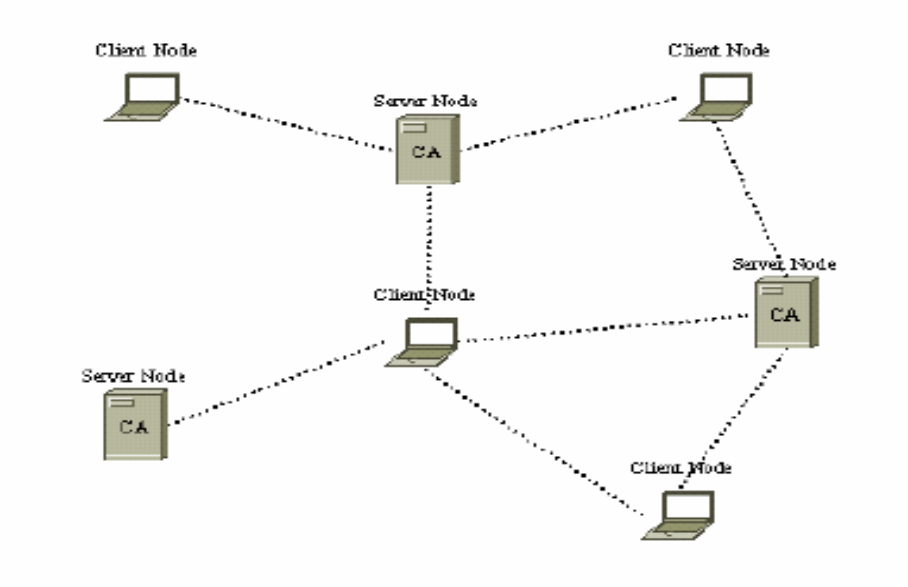

Fig. 3.2: Système contenant trois nœuds serveurs dont un est également un combinateur.

3.2.3 Utilisation de la cryptographie à seuil

Le partage de secret dans le service de gestion de clés est effectué en utilisant *la cryptographie à seuil* (*threshold cryptography*), voir chapitre 2. Un schéma à seuil (*t+1, n*) permet de distribuer les services[2] de l'autorité de certification à un groupe de *n* nœuds serveurs, de façon à ce que *(t+1)* nœuds puissent exécuter ensemble les services de l'autorité de certification, tandis qu'il est impossible d'exécuter ces services par au plus *t* nœuds **[CAP 03, ZHO 99].**

Chacun de ces nœuds serveurs est capable de générer une signature partielle en utilisant sa propre part de la clé privée de signature (s_i). Mais pour obtenir un certificat valide, il faut combiner *(t+1)* signatures partielles.

Dans ce système, les *n* nœuds serveurs du service de gestion de clés partagent l'aptitude de signer les certificats. Chaque nœud serveur génère une signature partielle pour le certificat en utilisant sa clé privée partagée, ensuite il envoie la signature partielle à un nœud combinateur. Avec (*t+1*) signatures partielles correctes, le combinateur est capable de calculer la signature du certificat.

[1] Les nœuds clients peuvent envoyer des requêtes *query* pour avoir les clés publiques des autres clients ou des requêtes *update* pour changer leurs clés publiques.

[2] Un service de l'autorité de certification peut être une génération de clé publique, une création d'un certificat numérique, …

Cependant, les nœuds serveurs compromis (au plus *t* d'entre eux) ne peuvent pas générer des certificats corrects ; car ils peuvent générer au plus *t* signatures partielles (voir figure 3.3).

L'utilisation de la cryptographie à seuil (*t+1, n*), assure que le service peut tolérer un certain nombre *t<n* de serveurs compromis. La figure 3.3 montre la génération des signatures en utilisant le schéma cryptographique à seuil (2, 3). Pour un message m, chaque serveur i peut générer une signature partielle PS (m; s_i) utilisant sa part s_i. Les deux serveurs 1 et 3 génèrent chacun une signature partielle et l'envoie au combinateur c. Même si le serveur 2 ne crée pas une signature correcte, le combinateur c sera capable de générer une signature de m signée par la clé privée *k*.

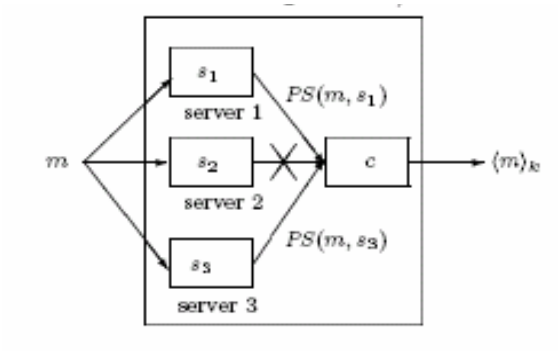

Fig. 3.3 : Exemple d'un schéma de cryptographie à seuil (2, 3)

Le système doit être protégé contre les serveurs compromis. Par exemple, un serveur compromis peut générer une signature partielle incorrecte, l'utilisation de cette signature partielle va produire une signature invalide. Heureusement, le combinateur peut vérifier la validité de la signature calculée, en utilisant les services de la cryptographie à clé publique (chapitre 2).

Pour s'assurer qu'un combinateur compromis ne peut pas prévenir la création d'un certificat valide, on peut utiliser (*t+1)* nœuds serveurs comme combinateurs pour s'assurer qu'au moins un combinateur est correct et qu'il peut calculer une signature valide **[ZHO 99]**.

Des schémas efficaces pour combiner les signatures partielles, sont proposés dans **[GEN 96, JAR 96]**. Ces schémas exploitent les redondances dans les signatures partielles, sachant que (*t+1*) signatures partielles correctes contiennent toute l'information sur la signature finale valide. Ces schémas utilisent aussi les codes de correction d'erreurs pour masquer les signatures partielles incorrectes.

3.2.4 La gestion des certificats

Création de certificat

Pour qu'un nœud puisse joindre le réseau, il doit d'abord obtenir un certificat valide de l'autorité de certification.

Renouvellement de certificat

Les certificats sont valides seulement pour une période donnée, donc il faut les renouveler avant l'expiration du temps. Lorsqu'un nœud veut renouveler son certificat, il doit envoyer une demande de renouvellement de certificat auprès de *(t+1)* nœuds serveurs au minimum. Si la demande est accordée, chacun de ces *(t+1)* nœuds serveurs génère un certificat partiel avec une nouvelle date d'expiration. Ces certificats partiels sont ensuite envoyés à un combinateur, qui pourrait être un des *(t+1)* serveurs. Dans le cas où un de ces *(t+1)* serveurs serait compromis, le certificat partiel envoyé au combinateur est invalide. Ainsi le certificat produit par le combinateur est aussi invalide. Quand le combinateur détecte que le certificat produit est invalide, il demande un autre ensemble de t+1 certificats partiels jusqu'à l'obtention d'un certificat valide.

Si un nœud veut changer sa paire de clé privée / publique, il doit mettre à jour son certificat associé à la nouvelle clé publique, cette opération est effectuée de la même manière que le renouvellement d'un certificat.

Demande des certificats

Les nœuds serveurs sont responsables de sauvegarder les certificats de tous les nœuds du réseau. Ceci permet à un nœud qui veut acquérir une clé publique des autres nœuds de demander le certificat correspondant à partir de n'importe quel nœud serveur. Pour cette raison, ce service exige que tous les nœuds doivent enregistrer leurs certificats dans les nœuds serveurs quand ils rejoignent au départ le réseau. Les nœuds serveurs doivent également avoir un mécanisme pour synchroniser leurs annuaires de certificat dans le cas des mises à jour et du renouvellement.

Maintenance du système

La maintenance de l'autorité de certification implique deux fonctions ; la *distribution des parts initiales* et la *mise à jour de ces parts* afin de se protéger contre les attaques. La mise à jour des parts peut aussi permettre au système de changer sa configuration, par exemple d'un système à seuil (3, 8) à un système à seuil (2, 5). Ce changement est utile si les serveurs sont compromis ou s'ils sont indisponibles pour d'autres raisons.

La distribution des parts initiales est effectuée par une entité administrative appelée « *dealer* », qui doit générer *n* parts de la clé privée de l'autorité de certification CA, ensuite elle fournit chacun des *n* nœuds serveurs avec une part.

A des intervalles périodiques, les nœuds serveurs peuvent mettre à jour leurs parts de la clé privée. Chaque nœud serveur calcule des nouvelles parts à partir des parts précédentes et envoie une part à chacun des autres nœuds serveurs (figure 3.4). Les nouvelles parts constituent donc un nouveau schéma (*t+1, n*) de partage de la clé privée. Cette opération est effectuée en utilisant les schémas proactifs (proactive schemes), pour plus de détails mathématiques sur ces schémas, voir **[JAR 95, HER 95].**

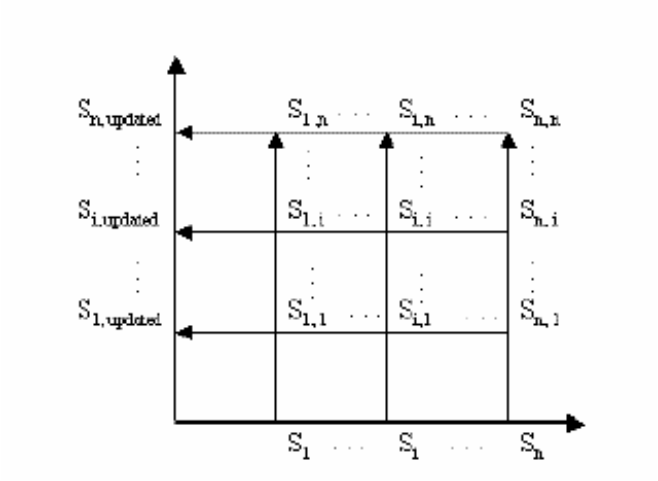

Fig.3.4 : Illustration du mécanisme de mise à jour de parts.

3.2.5 Analyse de la solution

Cette solution est convenable pour les réseaux ad hoc de long terme, qui ne sont pas spontanés. Cependant, elle exige que tous les nœuds soient capables de faire les calculs nécessaires, du fait que cette solution est basée sur le chiffrement à clé publique qui nécessite une grande puissance de calcul.

La sécurité de ce système dépend du nombre total *(t+1)* des nœuds serveurs. Si *t+1=1*, c'est le cas d'une seule autorité centralisée, qui est totalement vulnérable. Si *t+1=n*, la sécurité du système est compromise une fois que n'importe quel nœud est attaqué, ce qui est également vulnérable. Au moins *N = 3*t + 1* est suggéré dans **[ZHO 99]**. Mais on ne peut pas savoir si le seuil est suffisamment adéquat dans différents environnements ad hoc.

Cette solution présente un inconvénient : est que le système suppose que certains nœuds doivent jouer le rôle des serveurs, ce qui n'est pas toujours réaliste, au moins dans les applications civiles.

Un autre problème est comment rétablir le secret partagé ; car les *(t+1)* nœuds choisis peuvent être changés à cause de la topologie dynamique du réseau ou à cause des nœuds compromis. Cela est difficile à réaliser à cause de l'absence d'une autorité centralisée qui gère les nœuds serveurs.

3.3 L'autorité de certification entièrement distribuée

Cette solution est d'abord proposée par Luo et Lu dans **[LUO 00]**, ensuite elle est analysée par Luo et al dans **[KON 01]** et dans **[LUO 02].** Un schéma de cryptographie à seuil (*t, n*) est utilisé afin de partager la clé privée de signature parmi tous les nœuds du réseau tel que chaque nœud possède une part de la clé de signature. Puisque la clé privée est distribuée parmi tous les nœuds, on n'aura pas besoin d'élire ou de choisir les nœuds serveurs (servers).

3.3.1 Description du système

Dans cette solution, les services[1] de l'autorité de certification (CA) sont distribués à tous les nœuds du réseau ad hoc (figure 3.5). Les opérations qui nécessitent l'utilisation de la clé privée de l'AC ne peuvent être exécutées que par la collaboration de *t* nœuds ou plus.

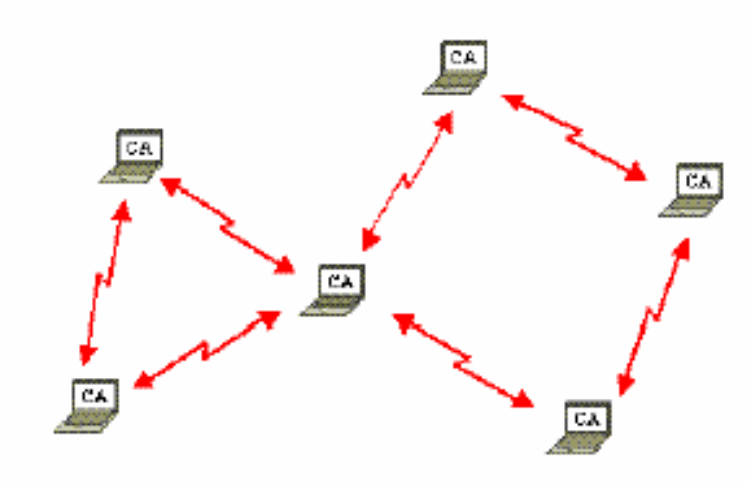

Fig. 3.5 : L'autorité de certification entièrement distribuée où tous les nœuds du réseau sont égaux et chacun possède une partie de la clé de signature.

Les services fournis par cette autorité de certification CA, peuvent être des services de gestion des certificats ou des services de maintenance du système. Les services de gestion des certificats incluent le renouvellement et la révocation des certificats, tandis que les services de maintenance permettent d'intégrer les nœuds qui veulent joindre le réseau, dans l'autorité de certification CA. Les services de maintenance fournissent donc chaque nœud par une part de la clé privée du CA. Ce service est appelé *service d'initialisation de parts* (*share initialization*). Le système de maintenance inclut aussi la mise à jour de ces parts pour les protéger des attaques. Ce service est dit *service de mise à jour de parts (share update)*. La disponibilité de ce service est basée sur la supposition que chaque nœud a au moins *t* nœuds voisins, et qu'il est fournit avec un certificat valide, avant qu'il joigne le réseau. Le système fournit ensuite des services pour maintenir et mettre à jour les certificats initiaux.

3.3.2 La maintenance du système

Dans ce paragraphe, on décrit les étapes nécessaires pour initialiser et maintenir le service distribué de l'autorité de certification CA. La maintenance est nécessaire pour manipuler les nouveaux nœuds qui veulent joindre le réseau et pour protéger le service du CA contre les attaquants qui essayent de compromettre le système.

[1] Les services de l'autorité de certification CA incluent les services de création, de renouvellement et de révocation des certificats.

3.3.2.1 Initialisation du système

Durant la phase d'initialisation du système (bootstrapping phase), une autorité administrative responsable du réseau ad hoc (dite « *dealer* ») doit initialiser les *k* nœuds premiers. L'initialisation consiste à fournir les nœuds avec leurs certificats, de certificat de l'autorité de certification et leurs parts de la clé privée de l'autorité de certification. Cette phase est illustrée dans la figure (3.6).

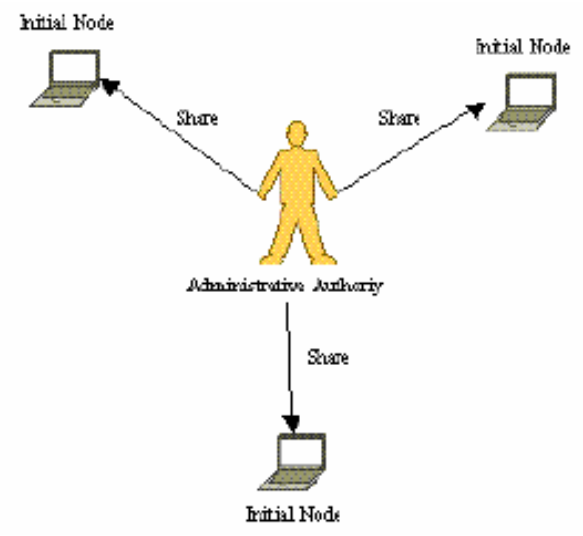

Fig. 3.6 : Distribution des parts durant la phase d'initialisation.

L'entité « dealer » est la seule entité qui a l'accès ou la connaissance de la clé privée de signature de CA. Cette entité peut par conséquent délivrer les certificats initiaux, comme c'est déjà mentionné. Après la phase d'initialisation des *t* nœuds premiers, l'entité « dealer » sera responsable seulement de l'enregistrement et de la certification initiale (c'est à dire la création des certificats) des nouveaux nœuds voulant joindre le réseau.

3.3.2.2 Initialisation des parts

Chaque nœud voulant joindre le réseau est impliqué dans l'autorité de certification distribuée CA, en lui fournissant une part de la clé de signature du CA. Puisque l'entité « dealer » n'est plus une partie du réseau, le mécanisme de distribution des parts est manipulé par les nœuds qui sont déjà initialisés (figure 3.7).

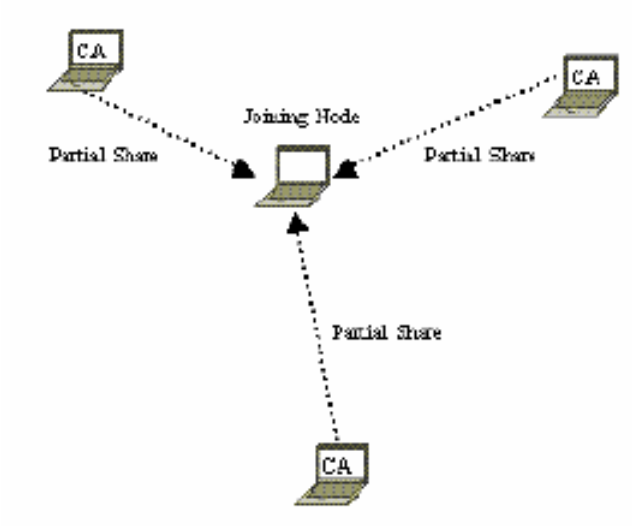

Fig. 3.7 : Initialiser le nœud qui veut joindre le réseau avec une nouvelle part.

Les nœuds qui composent le service du CA doivent générer une nouvelle part à partir de leurs parts. Chacun de ces nœuds déjà initialisés génèrent une sous part (partiel share) pour le nouveau nœud. En combinant les *t* sous parts, on obtient la *part* du nouveau nœud.

Cependant, le nœud qui joint le réseau ne peut pas savoir les valeurs des *t* sous parts, mais il connaît seulement le résultat de la combinaison des sous parts (c'est à dire, il connaît seulement la valeur de sa part).

Pour protéger les parts secrètes des nœuds, ces derniers font brouiller (shuffle) leurs sous parts (partiel shares) avant de les envoyer au nœud qui veut joindre le réseau. La figure 3.8 illustre le mécanisme qui fait brouiller les parts dans le cas où le réseau contiendrait 2 nœuds, le nœud 1 et le nœud 2. D'abord, les nœuds 1 et 2 se mettent d'accord d'une manière sécurisée sur un facteur aléatoire $d_{1,2}$ où un des deux nœuds traite ce facteur comme un nombre positif et l'autre le traite comme un nombre négatif. Les deux nœuds ajoutent ensuite le facteur à leurs sous parts et envoient le résultat brouillé des sous parts au nœud 3 qui veut joindre le réseau. Le nœud 3 ensuite fait la somme des sous parts brouillées ayant pour résultat sa part complète. Dans le cas où *t* serait plus grand que 2, c'est à dire, le réseau se compose de plus de 2 nœuds, chaque paire de nœuds {i, j} dans le réseau se met d'accord sur un facteur $d_{i,j}$. Par conséquent, le nombre de facteurs qui doivent être distribués est $t\,(t-1)/2$, où *t* est le nombre de nœuds.

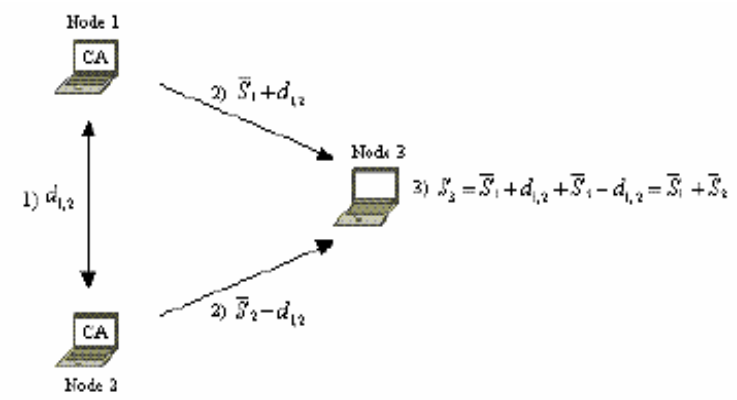

Fig. 3.8 : Schéma montrant le fonctionnement de brouille des sous parts

Après l'initialisation d'un nœud avec sa part de la clé de signature du CA, il devient un membre de l'autorité de certification et peut ainsi participer aux services du CA. Il peut donc renouveler, révoquer un certificat ou initialiser les nœuds qui veulent joindre le réseau avec leur part de la clé de signature du CA.

3.3.2.3 Renouvellement des parts

Pour protéger les parts contre les attaquants qui peuvent compromettre *t* nœuds ou plus

Un attaquant peut compromettre des nœuds en lui laissant suffisamment de temps, il peut obtenir ainsi leurs parts, lui permettant de reconstruire le secret (dans notre cas le secret est la clé de signature du CA). Pour se protéger contre de telles attaques, il est nécessaire d'utiliser un schéma proactif de partage de secret. Ces schémas vont mettre à jour les parts des nœuds de façon régulière, pour plus des détails sur ces schémas, voir **[HER 95, JAR 95]**.

3.3.2.4 Création des certificats

L'autorité de certification distribuée CA ne délivre pas de nouveaux certificats. Elle gère seulement les certificats qui ont été initialement créés. La responsabilité d'initialiser, d'enregistrer et de délivrer les certificats pour les nouveaux nœuds appartient à l'entité administrative « dealer », qui est responsable de l'installation du système.

3.3.2.5 Renouvellement des certificats

Un certificat n'est valide que pour une période de temps limitée. De ce fait, il est nécessaire de le renouveler avant qu'il arrive à sa date d'expiration. Quand un nœud *p* veut renouveler son certificat, il envoie une requête de renouvellement à un groupe de *t* nœuds voisins. Chaque nœud de ce groupe vérifie d'abord que la date de l'ancien certificat n'a pas été expirée et que le certificat lui-même n'a pas été révoqué. Si les nœuds voisins acceptent de servir la requête, chacun d'eux doit délivrer un nouveau certificat partiel, ensuite il doit l'envoyer au nœud *p* (figure3.9).

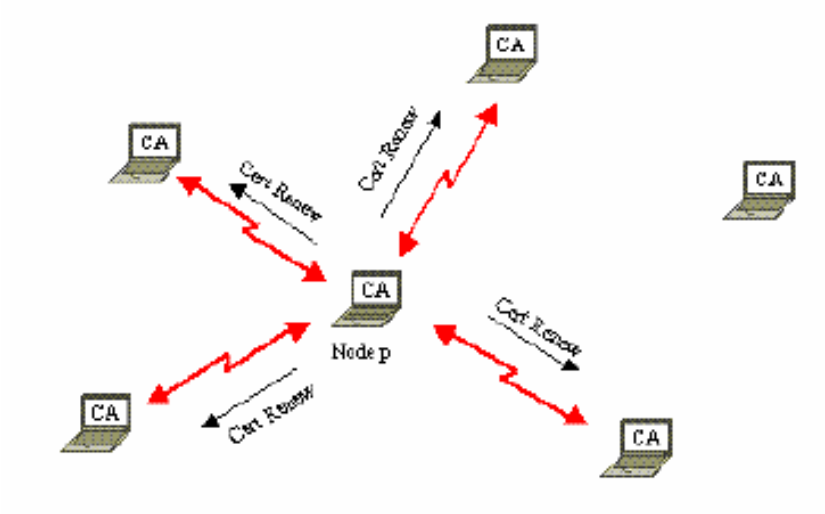

Fig. 3.9 : Une requête de renouvellement de certificats

3.3.2.6 Révocation de certificat

Le mécanisme de révocation de certificat est basé sur la supposition que tous les nœuds du réseau peuvent surveiller le comportement de leurs voisins (les voisins d'un seul saut) et peuvent maintenir leurs propres listes de révocation CRL. Si un nœud découvre qu'un de ses voisins est mal veillant (ou un attaquant), il doit le mettre dans sa liste de révocation de certificat CRL, ensuite il diffuse une accusation contre le nœud. Chaque nœud recevant cette accusation, doit d'abord vérifier dans sa liste de révocation, que l'accusation n'est pas diffusée par un nœud dont le certificat est révoqué. Si le certificat de celui qui a envoyé l'accusation a été déjà révoqué, l'accusation sera annulée. Quand un seuil des accusations légitimes (c'est à dire : *t* accusations) est reçu contre un nœud, le certificat de ce nœud doit être révoqué.

La figure (3.10) illustre les séquences d'événements exécutés, dans le cas où un nœud mal veillant D serait détectée par les deux nœuds B et C. D'abord ces deux nœuds révoquent le certificat du nœud D, ensuite chacun d'eux diffuse une accusation. A la réception des accusations, le nœud A doit vérifier sa liste de révocation CRL pour s'assurer que les certificats des nœuds B et C n'ont pas été révoqués. Puisque le nœud A a reçu un seuil d'accusation (dans cet exemple $t = 2$), il doit mettre le nœud D dans sa liste de révocation CRL. Le nœud A retransmet les accusations des nœuds B et C. En recevant ces accusations, les nœuds E et F exécutent les mêmes étapes que le nœud A.

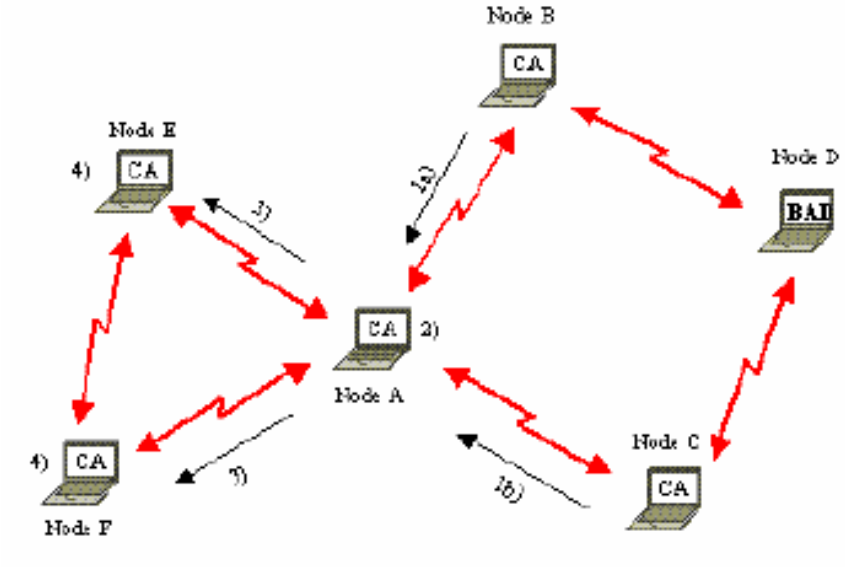

Fig. 3.10 : La révocation d'un certificat et la maintenance des listes de révocation de certificats.

3.3.3 Analyse de la solution

Similairement à la solution précédente de l'autorité de certification partiellement distribué, cette solution convient aussi aux réseaux ad hoc de long terme et qui ne sont pas spontanés, avec des nœuds qui possèdent une grande puissance de calcul, afin d'exécuter les opérations de la cryptographie à clé publique.

L'avantage principal de cette solution est qu'elle est disponible, et à la différence des solutions basées sur les certificats, elle fournit un mécanisme de révocation de certificat.

Le coût pour réaliser cette disponibilité élevée est l'utilisation d'un ensemble de protocoles de maintenance très complexes (par exemple : les protocoles d'initialisation des parts et les protocoles de mise à jour des parts). De plus, un grand nombre de parts est exposé aux attaques, puisque chaque nœud a sa propre part, à la différence de la solution partiellement distribuée. Par conséquent, le paramètre de seuil *t* doit être choisi[1] plus grand puisqu'un attaquant peut compromettre un grand nombre de parts.

La méthode de révocation de certificats suppose que chaque nœud est capable de surveiller le comportement de ces nœuds voisins (les voisins d'un seul saut), par exemple en surveillant le trafic du réseau. Cependant cette supposition est difficile dans certains réseaux ad hoc.

[1] Le paramètre *t* doit être prudemment choisi, de façon à ce que l'attaquant ne puisse pas compromettre *t* parts avant l'exécution de la procédure de mise à jour des parts.

3.4 Certificats délivrés individuellement (Self issued certificates)

3.4.1 Introduction

Cette approche est proposée par Hubaux et al dans **[HUB 01]** et fournit une solution de gestion de clés publiques similaire à la solution PGP (chapitre 2), dans le sens où les certificats sont délivrés par les utilisateurs eux-mêmes, en se basant sur leurs connaissances personnelles, sans l'implication d'aucune autorité de certification **[CAP 03]**.

3.4.2 Principe de la solution

Cette solution, comme le PGP, traite le problème de distribution de clés publiques de façon à garantir l'authentification. Contrairement aux solutions PKI traditionnelles, dans la solution PGP, les clés publiques ne sont pas certifiées par une entité digne de confiance, par exemple l'autorité de certification. Mais chaque utilisateur possède la capacité de certifier les clés publiques des autres utilisateurs. Chaque utilisateur détermine alors la valeur de confiance à placer dans un certificat spécifique.

La figure 3.11 montre un exemple simple de fonctionnement de PGP, par exemple Bob délivre un certificat à Chris cela prouve que *pkChris* est réellement la clé publique appartenant à Chris. Alice a aussi délivré un certificat à Bob, indiquant que *pkBob* est réellement la clé publique appartenant à Bob. Alice fait confiance également à Bob pour ne délivrer aucun certificat faux, ainsi Alice fera confiance à tous les certificats délivrés par Bob. Par conséquent avoir le $cert_{Alice,\ Bob}$ et $cert_{Bob,\ Chris}$, Alice peut vérifier que la clé publique de Chris, pkChris est authentique. Elle peut donc communiquer avec Chris en toute sécurité, même s'ils ne sont jamais rencontrés.

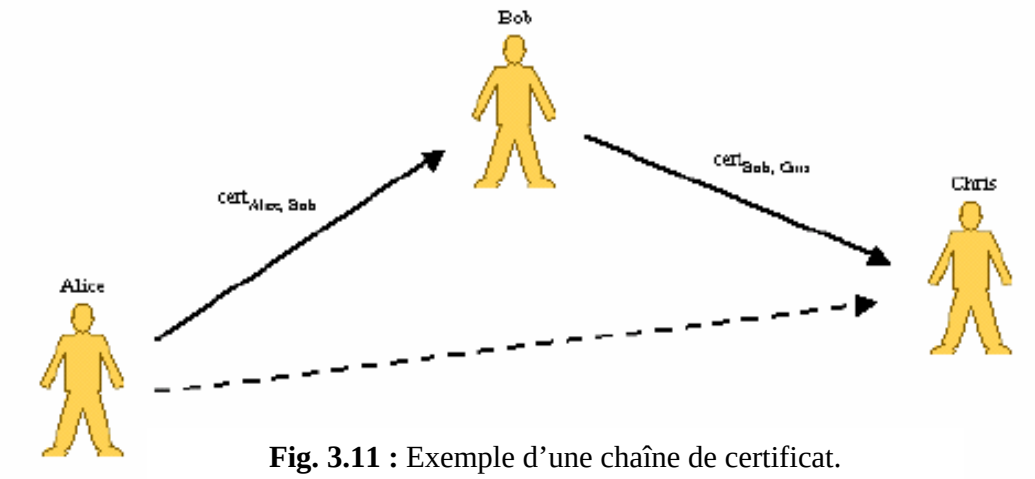

Fig. 3.11 : Exemple d'une chaîne de certificat.

3.4.3 Les chaînes de certificats

Dans PGP, les serveurs de clés publiques (c'est à dire les annuaires de certificat) sont utilisés pour distribuer les certificats. Cependant, dans les réseaux ad hoc, les serveurs ne sont pas disponibles et par conséquent, dans la solution proposée par Hubaux et al, les utilisateurs eux-mêmes sont responsables de distribuer et de sauvegarder les certificats. Chaque utilisateur possède un répertoire local de certificats dont lequel il sauvegarde un nombre restreint de certificats qui ont été délivrés. Quand deux utilisateurs souhaitent vérifier leurs clés publiques, ils essayent de trouver une chaîne de certificat en combinant seulement les certificats contenus dans leurs répertoires locaux, voir figure 3.12 **[CAP 03, FOK 02]**.

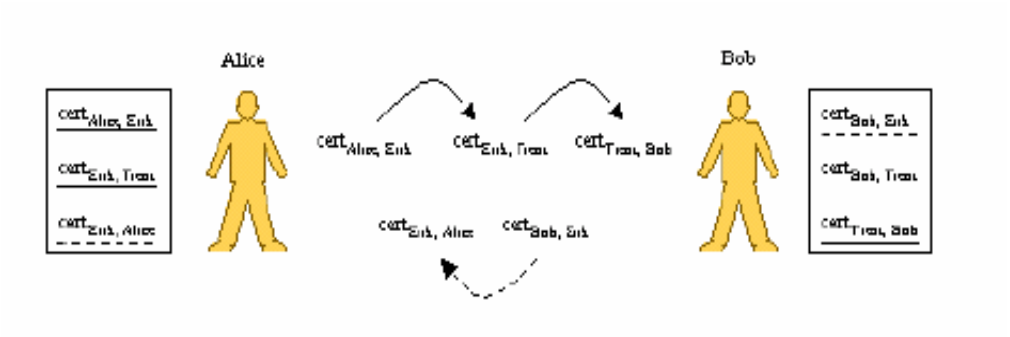

Fig. 3.12 : Création d'une chaîne de certificats utilisant seulement les certificats sauvegardés localement.

3.4.4 Graphe de modélisation

Les relations entre les nœuds de ce système sont représentées par les certificats des clés publiques. Ces relations peuvent être modélisées par un graphe orienté G (V, E), où V et E représentent respectivement les ensembles des sommets et les ensembles des arcs de ce graphe. Les sommets représentent les nœuds tandis que les arcs représentent les certificats des clés publiques. Précisément, si on trouve dans le graphe un arc orienté du sommet *u* vers le sommet *v*, cela signifie que le nœud *u* a délivré un certificat au nœud *v*, (voir figure 3.13).

Une chaîne de certificat du nœud *u* vers le nœud *v* est représentée par un chemin orienté du sommet *u* vers le sommet *v* dans le graphe G **[HUB 01]**.

Comme nous avons déjà mentionné, chaque nœud possède un répertoire local des certificats. Ce répertoire contient les certificats qu'un nœud a délivrés et d'autres certificats sélectionnés que les autres nœuds ont délivrés. Pour notre modélisation, cela signifie que chaque utilisateur *u* sauvegarde les arcs sortants (avec leurs sommets correspondants) à partir du sommet *u*, et un autre ensemble des arcs sélectionnés (avec leurs sommets correspondants) de ce graphe. L'ensemble des arcs (et sommets) sélectionnés est un sous graphe qui appartient à *u*.

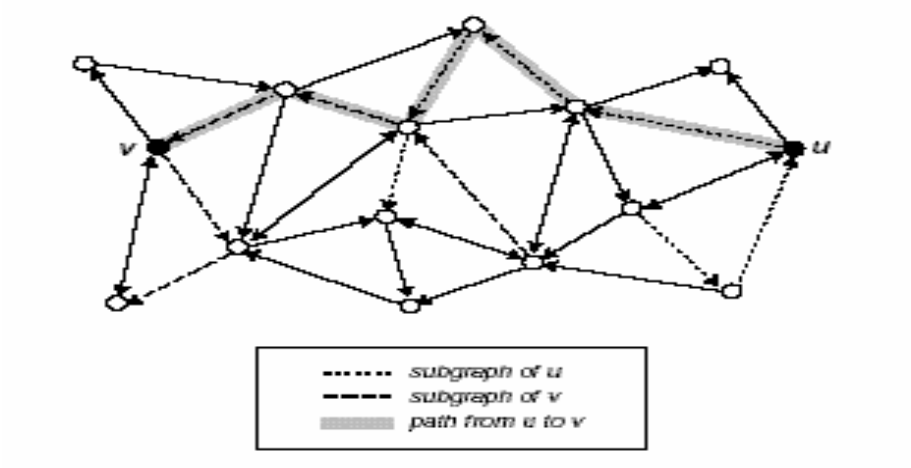

Fig. 3.13 : La fusion des sous graphes.

Quand un nœud *u* veut vérifier la clé publique du nœud *v*, *u* et *v* fusionnent leurs répertoires des certificats sélectionnés, ensuite *u* essaie de trouver une chaîne de certificats du nœud *u* vers le nœud *v* dans le répertoire fusionné. Pour cette modélisation, *u* et *v* fusionnent leurs sous graphes, ensuite *u* essaie de trouver un chemin du sommet *u* vers le sommet *v* dans le sous graphe fusionné. Un exemple est illustré dans la figure 3.13.

3.4.5 Analyse

L'avantage principal de cette solution est qu'elle ne nécessite aucune forme d'infrastructure **[FOK02]**. Les utilisateurs eux-mêmes gèrent leurs clés.

Les auteurs de cet article ont proposé plusieurs algorithmes de construction de chaînes de certificats et ont montré que même des algorithmes simples peuvent atteindre des performances élevées, dans le sens où chaque utilisateur peut trouver au moins une chaîne de certificat (avec une grande probabilité) pour n'importe quel utilisateur, même dans le cas où la taille des répertoires locale des utilisateurs serait petite.

Contrairement aux solutions précédentes, basées sur les clés publiques, cette solution est mieux destinée à fonctionner dans les réseaux ad hoc spontanés, où les nœuds n'ont aucune relation de confiance préalable **[FOK 02].** Cependant, une phase initiale est nécessaire et par conséquent la solution sera limitée et ne conviendra pas aux réseaux ad hoc de court terme. Puisque ce système est basé sur le chiffrement à clé publique, les nœuds doivent avoir une capacité de calcul suffisante.

On cite encore un autre inconvénient : chaque utilisateur doit construire un répertoire local de certificat avant qu'il puisse utiliser le système, ce qui mène à une grande consommation en terme de temps et de bande passante **[HUB 03].**

3.5 Accord de clé dans un groupe (Key agreement)

Une approche différente proposée par Asokan et Ginzboorg **[ASO 00]** est basée sur un mot de passe partagé, connu par un groupe de nœuds. Comme mentionné par les auteurs, les nœuds qui veulent établir une session sécurisée doivent partager un secret.

3.5.1 Introduction

Dans le scénario considéré, un groupe de personnes se réunit dans une salle et forme un réseau sans fil avec leurs ordinateurs portables, pour la durée de la réunion. Les membres de cette réunion n'ont pas accès à une infrastructure de clés publiques ou à un service de gestion de clés. Donc, ils n'ont aucun moyen d'authentifier les autres membres.

Dans cet article, les auteurs ont décrit et introduit des méthodes d'échange de clés qui sont basé sur un mot de passe faible.

3.5.2 Description de la solution

L'idée principale de la solution proposée est la suivante : Un mot de passe est choisi et partagé par les participants dans la salle (par exemple, en l'écrivant sur un tableau). Cependant, ce serait une erreur d'employer ce mot de passe directement comme une clé partagée, car ce protocole serait alors vulnérable aux attaques de dictionnaire **[SCH 96]**. Par conséquent, chaque participant contribue une partie de la clé et signe cette donnée en utilisant le mot de passe faible. Ainsi, une clé forte (strong key) est dérivée à partir seulement d'un secret faible (c.-à-d., le mot de passe).

Cette opération est réalisée en utilisant une extension du protocole Hypercube, qui est basé sur le protocole d'échange de clés Diffie-Hellman (DH) décrit dans le chapitre 2.

3.5.3 Le protocole Hypercube

Les nœuds qui participent dans ce protocole sont arrangés de manière à ce que chaque nœud, représente un sommet dans un cube de dimension *d*, appelé hypercube. Ce protocole exécute *d* rondes (ou phases) du protocole d'échange de clés Diffie-Hellman. Durant chaque ronde $j = 1, d$, un nœud exécute le protocole biparti d'échange de clé avec son voisin dans la $j^{\text{ième}}$ dimension

Dans la première ronde, chaque nœud *i* utilise son propre secret x_i comme exposant. Dans les rondes suivantes, la clé obtenue à partir de la ronde précédente est utilisée comme un exposant secret.

La figure 3.14 illustre le fonctionnement du protocole Hypercube, où le nombre de participants est quatre, c'est à dire $d = 2$. Dans la première ronde, les nœuds *1* et *2* exécutent le protocole biparti (two-party) d'échange de clés DH, en parallèle les nœuds *3* et *4* exécutent le même protocole. Après la première ronde, les paires de nœuds (*1, 2*) et (*3,4*) partagent un secret commun $k_{1,2}$ et $k_{3,4}$ respectivement. Dans la deuxième ronde (la dernière dans ce cas), les nœuds *1* et *3* exécutent le protocole

biparti d'échange de clé DH, ainsi que les nœuds *2* et *4*. La clé échangée dans la ronde précédente est utilisée dans cette ronde comme un exposant dans le protocole d'échange de clé DH. Le nœud *1* envoie $g^{k1,\,2}\ mod\ p$ au nœud *3* et le nœud *3* envoie $g^{k3,\,4}\ mod\ p$ au nœud *1* Lorsque la deuxième ronde est terminée, les quatre nœuds partagent le même secret $k = k_{1,\,2,\,3,\,4}$.

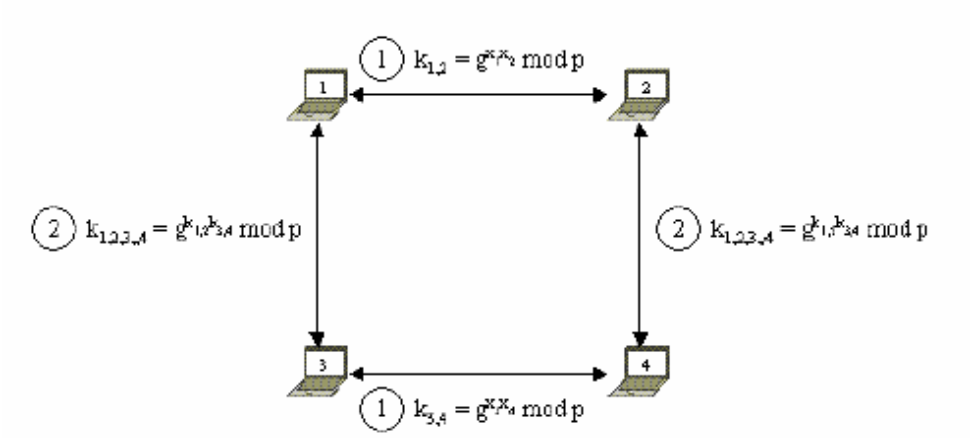

Fig. 3.14 : Le protocole Hypercube utilisé pour partager un secret commun $k = k_{1,\,2,\,3,\,4}$ entre quatre nœuds.

Ce protocole peut être généralisé à un groupe qui contient 2^d membres, ces membres peuvent être arrangés dans un cube de dimension *d*, ensuite les mêmes étapes décrites précédemment seront exécutées. Les auteurs du protocole original Hypercube ont aussi proposé un autre protocole appelé 2^d Octopus, ce protocole peut manipuler le cas où le nombre de membres dans un groupe n'est pas une puissance de *2*. Pour plus de détails voir l'article **[BEC 98]**. Les extensions du protocole Hypercube présentées par Asokan et Ginzborg dans l'article **[ASO 00]**, manipulent aussi le cas où le nombre de membres dans un groupe n'est pas une puissance de *2*.

3.5.4 Le protocole d'échange de clé (EKE)

Le protocole d'échange de clé EKE (Encrypted Key Exchange) fournit une clé DH authentifiée, basé sur un mot de passe. Les deux membres qui veulent échanger une clé cryptographique forte (strong key) se mettent d'accord sur un mot de passe simple *p*. Utilisant ce mot de passe, les deux membres chiffrent ensuite leurs valeurs publiques DH, utilisant le mot de passe partagé, ensuite chacun d'eux l'envoie à l'autre membre. Le nœud qui reçoit ce résultat, peut ensuite déchiffrer la valeur publique et calcule la valeur secrète *k*. le protocole procède comme suit :

1. $A \rightarrow B: A,\ E_p(g^{x_A} mod\ p)$
2. $B \rightarrow A: E_p(g^{x_B} mod\ p),\ E_k(c_B)$
3. $A \rightarrow B: E_k(c_A, c_B)$
4. $B \rightarrow A: E_k(c_A)$

Après qu'il reçoit le message dans l'étape 1, le nœud B peut extraire la valeur $g^x_A mod\ p$ et calcule la clé partagée k par $k = (g^x_A)^{x}_B\ mod\ p$. De la même manière A peut calculer la clé partagée après la réception du message envoyé dans l 'étape 2. Pour vérifier que les deux parties ont la même vue sur la clé partagée, c'est à dire les deux parties connaissent le mot de passe simple p et peuvent déchiffrer les messages dans les étapes 1 et 2, ils échangent des 'challenges' c_A et c_b.

Asokan et Ginzborg suggèrent l'utilisation du protocole EKE au lieu d'utiliser le protocole d'échange de clé DH dans chaque ronde du protocole Hypercube.

3.5.5 Analyse

Le système peut facilement être pénétré par un adversaire qui pourrait obtenir le mot de passe en utilisant un microphone ou une caméra.

Le mot de passe doit être partagé par tous les nœuds impliqués dans le réseau ad hoc, les auteurs n'ont pas pris en considération le renouvellement du mot de passe dans le cas où un nœud serait compromis ou quitte le réseau.

C'est une solution orientée groupe du fait qu'elle ne permet pas l'authentification de chaque nœud individuellement. Un autre aspect qui n'est pas précisé dans cette solution est le cas des nœuds qui veulent joindre ou quitter le réseau. Considérons par exemple une réunion où il manque un seul membre. Les membres présents commencent la réunion et effectuent l'échange de clé. Quand le membre absent arrive, il doit joindre le réseau. De ce fait, un mécanisme pour lui fournir une clé de chiffrement doit être présent. De même un membre qui quitte une réunion ne devrait pas pouvoir écouter de l'extérieur du lieu de réunion. Une méthode pour mettre à jour la clé de chiffrement lors du départ d'un membre, devrait également être disponible.

Le protocole Hypercube utilisé, suppose que les nœuds participant au système sont arrangés dans un hypercube.

3.6 Conclusion

Il est présenté dans ce travail, différentes approches, pour assurer l'authentification dans un réseau ad hoc. Chacune des approches présentées fournit un mécanisme de gestion de clé afin d'assurer l'authentification du réseau ad hoc.

La comparaison des solutions présentées n'est pas possible, puisqu'elles sont proposées pour différents types de réseau ad hoc. Cependant elles ont éliminé certaines difficultés qu'un réseau ad hoc présente. Les solutions présentées ont des problèmes qui doivent être résolues.

Nous allons proposer dans le chapitre suivant un protocole qui permet de résoudre le problème d'authentification dans les réseaux ad hoc et qui prend en considération les caractéristiques limitées des dispositifs mobiles.

Chapitre 4

Présentation d'un nouveau schéma d'authentification

Chapitre 4 : Présentation d'un nouveau schéma d'authentification

4.1 Introduction

Comme nous l'avons dit dans le chapitre précédent, le problème de l'authentification dans les réseaux ad hoc est compliqué parce que ces réseaux n'utilisent aucune infrastructure fixe, d'où la nécessité de concevoir des schémas ou des protocoles qui s'adaptent à ce nouvel environnement caractérisé par des fréquents déconnexions, changement de topologie et qui prend en compte les caractéristiques physiques des unités mobiles (vitesse CPU, bande passante limité, espace de stockage, source d'énergie limitée).

Différentes approches sont proposées dans le chapitre précédent, pour assurer l'authentification dans un réseau ad hoc. D'abord, l'autorité de certification partiellement distribuée qui utilise une infrastructure à clé publique distribuée. Dans cette approche un secret est partagé parmi un groupe de nœuds choisis parmi les nœuds du réseau. Ce groupe formera ensuite le service de gestion de clé distribué. Une deuxième approche : l'autorité de certification totalement distribuée qui est similaire à la première, mais qui partage le secret parmi tous les nœuds du réseau. Ces deux approches utilisent un schéma de cryptographie à seuil (threshold cryptography) pour distribuer les services de l'autorité de certification parmi les nœuds du réseau. La deuxième approche apporte beaucoup d'améliorations, mais elle est très complexe et le principal inconvénient de la première approche est son manque d'un mécanisme de révocation de certificats. Une autre approche (les certificats délivrés individuellement), contrairement aux deux premières, ne considère aucune autorité de certification. Chaque nœud dans cette approche possède la capacité de certifier les clés publiques des autres nœuds en se basant sur des connaissances personnelles, similairement à la solution PGP. Cette solution présente un avantage important est qu'elle ne nécessite aucune forme d'infrastructure, mais elle manque aussi d'un mécanisme de révocation de certificats. Une approche différente est basée sur un mot de passe partagé, connu par un groupe de nœuds. Pour cela des algorithmes d'échange de clés basés sur un mot de passe faible, sont utilisées dans cette approche (comme GDH, EKE, …). Ces algorithmes nécessitent une structure prédéfinie de la topologie du réseau. Cette solution est convenable pour les réseaux orienté groupe (comme les conférences). Cependant, il faut traiter le problème d'échange de mot de passe quand les nœuds quittent ou joignent le réseau.

L'étude de ces approches, nous a amené à conclure qu'il n'existe pas une approche spécifique qui assure l'authentification dans n'importe quel environnement ad hoc. En d'autres termes, on peut dire qu'il n'y a pas un modèle universel pour garantir l'authentification dans un réseau ad hoc. Le choix de la solution dépend des types de nœuds qui forment le réseau, et de degré de sécurité à assurer.

L'inconvénient commun entre ces différentes solutions est l'utilisation des algorithmes cryptographiques asymétriques (à clé publique) qui nécessite une grande puissance de calcul. Dans ce chapitre, Afin de vérifier l'authentification des nœuds dans un réseau ad hoc, nous avons essayé de minimiser le nombre des opérations de calcul. L'utilisation des algorithmes cryptographiques asymétriques tel

que RSA nécessite plus de calcul que les algorithmes cryptographiques symétriques tel que DES, qui nécessite plus de calcul que les fonctions de hachage tel que MD5 ou SHA. Dans notre solution, nous proposons d'utiliser les algorithmes asymétriques uniquement dans la phase d'initialisation, ensuite nous utilisons les chaînes de hachage (hach chains), qui sont basées sur les fonctions de hachage à sens unique.

Dans ce chapitre on va introduire d'abord les chaînes de hachage qui sont basées sur les fonctions de hachage à sens unique. Ensuite nous allons décrire les différentes phases de notre protocole et à la fin nous citons quelques inconvénients et avantages de la solution.

4.2 Les chaînes de hachages

4.2.1 Introduction

Les chaînes de hachage sont créées en utilisant les fonctions de hachage à sens unique. Une fonction de hachage est une fonction mathématique qui convertit une chaîne de caractères de taille quelconque en une chaîne de caractère de taille fixe (appelée empreinte), et qui possède les caractéristiques suivantes :

- Etant donné une chaîne d'entrée *x*, on peut calculer h (x). Où h (.) est la fonction de hachage.
- Etant donné une chaîne *y* aléatoirement choisie, il est impossible de trouver *x* tel que h(*x*) =*y*. (*caractéristique : sens unique*).
- Pour tout *x*, il est impossible de trouver *x'* tel que h (*x*) = h (*x'*). (*caractéristique résistance aux collisions*).

Donc une fonction de hachage à sens unique est une fonction dont il est facile de calculer l'empreinte à partir de la chaîne d'entrée mais il est presque impossible d'engendrer des chaînes qui ont une certaine empreinte. Aussi il est mathématiquement impossible de remonter à la chaîne d'origine à partir de l'empreinte. Pour toute fonction de hachage, la probabilité d'une collision[1] est estimée à 2^n où n est la taille de l'empreinte.

Les auteurs de **[EVA 74]** décrivent dans leur article comment une fonction de hachage peut garantir l'authentification d'un utilisateur. Un utilisateur calcule la sortie de la chaîne de hachage, *y= h(x)*, et partage cette valeur en toute sécurité avec le système ou l'utilisateur avec qui il veut s'authentifier. L'utilisateur est authentifié en révélant la valeur *x*, qui est connue seulement par lui puisque la valeur *y* ne peut pas être inversée du à la propriété « sens unique ».

4.2.2 Définition de la chaîne de hachage

Une *chaîne de hachage* de longueur N est construite en appliquant une fonction de hachage N fois sur une valeur aléatoire appelé x_N. La valeur x_N est appelée valeur *racine* de la chaîne de hachage. On définit une chaîne de hachage en utilisant la fonction de hachage *h* par :

$$\begin{cases} h_i(y) = h(h_{i-1}(y)) \\ h_0(y) = x_N \end{cases}$$

[1] Une fonction de hachage qui *résiste aux collisions* est une fonction dont il est difficile de trouver deux chaînes d'entrées différentes qui ont la même valeur de hachage (même empreinte).

Où $h_i(y)$ est le résultat de l'utilisation répétée i fois de la chaîne de hachage à la valeur initiale y. La valeur *finale* de hachage de la chaîne de hachage $x_0 = h_N(x_N)$ est obtenue en appliquant la fonction de hachage N fois.

On peut donc former la *chaîne de hachage* suivante de taille N :

$$h_1(y), h_2(y), h_3(y), \ldots, h_{N-1}(y), h_N(y).$$

En connaissant la valeur h_i, la valeur h_{i-1} ne peut pas être générée sans connaître la valeur *y*.

4.2.3 Application des chaînes de hachage

L'utilisation des éléments d'une chaîne de hachage pour assurer l'authentification est d'abord introduite en 1981 par Lamport **[LAM 81]**. Lamport propose l'utilisation répétée d'une fonction de hachage, pour générer une chaîne de valeurs permettant plusieurs authentifications d'un utilisateur.

Dans les schémas de chaînes de hachage, une fonction de hachage h (.) est appliquée N fois à une valeur aléatoire P_N, afin d'obtenir la valeur finale P_0 (figure 4.1).

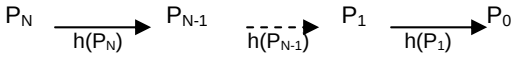

Fig. 4.1 : Génération d'une chaîne de hachage

Chaque valeur de hachage de la chaîne peut garantir une seule authentification de l'utilisateur. L'utilisateur envoie la valeur P_1 pour la première authentification, P_2 pour la deuxième authentification et ainsi de suite. Le récepteur applique une seule fois la fonction de hachage pour vérifier la valeur de hachage reçu. Puisque la fonction de hachage est à sens unique, seulement l'utilisateur qui a créé la chaîne de hachage peut générer la valeur de hachage qui précède la valeur envoyée.

L'utilisateur peut ne pas sauvegarder toutes les valeurs de la chaîne de hachage, il sauvegarde seulement la première valeur P_N (la racine) *racine* afin de pouvoir reconstruire la chaîne de hachage. Cette propriété est idéale pour les dispositifs qui sont caractérisés par une capacité limitée de stockage tels que les nœuds mobiles.

Weimerskirch et Westhoff proposent dans leur article **[WES 03]** un protocole qui utilise les chaînes de hachage pour assurer l'authentification et qui ne nécessite ni la présence d'une autorité de certification (CA) ni l'utilisation des certificats numériques. Le coût des calculs est basé sur le calcul des valeurs de hachages qui n'est pas cher. La valeur P_N de la chaîne de hachage est utilisée comme une clé privée de l'unité et la valeur finale P_0 comme une clé publique. Puisque cette solution ne suppose aucun canal sécurisé pour l'échange des clés publiques, les valeurs finales de hachage P_0 ne peuvent être échangées en toute sécurité. Ce schéma fournit ainsi une faible authentification orienté plus service que sécurité. Cette solution permet

aux unités de reconnaître l'unité qui a déjà fournit un service. Ainsi, la clé publique est liée à un service et non à une identité.

4.3 Description du système

On considère un système composé de *m* nœuds mobiles. Chaque nœud $_i$ dans le réseau ad hoc génère une chaîne de hachage : P_0^i, P_1^i, P_2^i, P_3^i, ..., P_N^i. Ensuite il sauvegarde la racine P_N^i en toute sécurité[1]. Dans notre système, la valeur P_N^i est considérée comme la clé privée du nœud $_i$ et la valeur P_0^i est considérée comme sa clé publique, comme dans le cas d'une infrastructure à clé publique (une PKI).

4.3.1 Supposition

- Chaque nœud $_i$ a un identificateur unique ID_i.
- Chaque nœud possède une paire de clés publique / privée générée en utilisant l'algorithme à clé publique RSA.
- Chaque nœud utilise une fonction de hachage à sens unique h (.). On utilise dans notre protocole la fonction cryptographique SHA-1 qui produit des empreintes digitales (des condensats) de taille 160 bits.
- Chaque nœud a un certificat numérique signé par une AC digne de confiance
- Les liens de communications sont fiables et bidirectionnels.
- On fixe la valeur N = 100, c'est-à-dire chaque utilisateur génère une chaîne de hachage de taille N = 100, ce qui permet de satisfaire les besoins d'un réseau ad hoc de taille et de durée moyenne **[**WES 03].

4.3.2 Buts du protocole

On décrit dans ce paragraphe les caractéristiques que doit vérifier un protocole d'authentification dans un réseau ad hoc pour être applicable :

- **Minimiser les opérations de calcul :** puisque les nœuds dans un réseau ad hoc sont alimentés par des sources d'énergie autonomes très limité, un protocole d'authentification doit exécuter un nombre minimum d'opérations de calcul.

- **Minimiser le coût du calcul :** il est préférable qu'un protocole d'authentification utilise des opérations de calcul de coût minimum, puisque les nœuds mobiles ont une puissance de calcul limitée. C'est pour cette raison que nous avons choisi l'utilisation des fonctions de hachage.

- **Le flux des messages :** puisque la transmission de messages nécessite une grande source d'énergie (les batteries), le nombre des messages échangées doit être le plus petit que possible.

- **La taille des messages :** puisque la bande passante réservée à un hôte dans un réseau sans fil est très petite. Les messages assez longs sont divisés en

[1] Il peut sauvegarder la racine seulement ou toute la chaîne de hachage.

plusieurs paquets. L'envoi de plusieurs paquets contredit la caractéristique précédente. Donc il est préférable d'envoyer des paquets de données de petite taille.

- **Mémoire de stockage de données :** due à la capacité limitée de mémoire, les protocoles d'authentification ne doivent pas nécessiter la sauvegarde de plusieurs clés cryptographiques ou de grande quantité d'information.

- **Limiter les conséquences de la révélation d'une information secrète :** les nœuds ad hoc fournissent un niveau bas de sécurité physique. Lorsqu'un attaquant accède aux informations d'un nœud, il peut obtenir toutes les données stockées. Cette attaque est possible du fait que les nœuds dans un réseau ad hoc ne peuvent pas être protégés de la même manière qu'un serveur dans une salle sécurisée, par exemple. Le protocole d'authentification devrait être conçu d'une manière que la révélation des données sauvegardées dans un nœud ne compromet pas le système entier. Quand tous les nœuds partagent la même clé symétrique, tout le système est compromis lorsqu'une clé est révélée. Donc les solutions qui utilisent différentes clés pour les différents nœuds sont plus convenables. C'est le cas de notre protocole où chaque nœud possède sa propre clé qui est représenté par une valeur de hachage.

4.3.3 Les étapes de l'authentification

Notre protocole **[ALI 04]** est constitué de deux phases essentielles. La première phase est la phase d'initialisation où une autorité de certification digne de confiance génère des paires de clés publique / privée et des certificats numériques pour les nœuds qui constituent le réseau, ensuite chaque nœud crée une chaîne de hachage et diffuse sa valeur finale de hachage à ses voisins d'un seul saut. Durant la deuxième phase (phase d'authentification), un nœud utilise une de ces valeurs de chaîne de hachage pour assurer son identité.

On décrit dans ce qui suit chacune des étapes nécessaires pour initialiser et assurer l'authentification dans un réseau ad hoc.

4.3.3.1 Phase d'initialisation

Première étape : ***Initialiser les nœuds avec les paires de clés et les certificats numériques***

Durant la phase d'initialisation, une autorité de certification[1] digne de confiance doit initialiser les nœuds qui forment le réseau ad hoc. L'autorité de certification génère pour chaque nœud $_i$ une paire de clés publique / privée (Pk_i, Sk_i) et le certificat associée $Cert_i$ (figure 4.2). L'autorité de certification ne doit pas être toujours en ligne comme dans le cas d'une infrastructure à clé publique (PKI) puisque on aura pas besoin de mette à jour les certificats ou de révoquer les certificats qui ne sont plus valides. Nous avons utilisé l'autorité de certification dans la phase d'initialisation afin d'échanger les valeurs finales P_0 des chaînes de hachage entre les nœuds qui

[1] L'autorité de certification est représentée par un nœud du réseau digne de confiance et responsable de l'initialisation des nœuds avec les paires de clés et les certificats numériques.

forment le réseau. L'utilisation des certificats numériques pour échanger les valeurs P_0 permet de s'assurer de l'identité des nœuds. De cette manière, une relation entre l'identité du nœud et la valeur finale P_0 est créée. L'autorité de certification sera utilisée par la suite pour initialiser les nœuds qui veulent joindre le réseau avec une paire de clés et un certificat numérique.

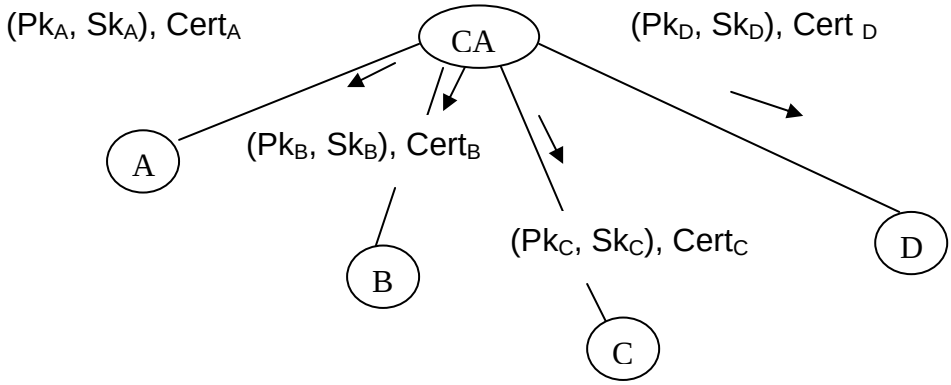

Fig. 4.2 : L'autorité de certification génère pour chaque nœud une paire de clés et un certificat numérique.

Deuxième étape : *diffusion des valeurs P0*

Chaque nœud $_i$ dans le réseau envoie sa valeur finale P_0^i, considérée comme une clé publique à tous les nœuds voisins d'un seul saut (one hop). Pour être sûr que c'est bien le nœud $_i$ qui a envoyé la valeur P_0^i et non pas un attaquant qui a intercepté le message qui contient la valeur P_0^i, le nœud $_i$ doit d'abord envoyer sa clé publique ainsi que le certificat numérique à tous les nœuds voisins d'un seul saut, ensuite il leurs envoie la valeur P_0^i chiffrée avec sa clé privée Sk_i.

Considérons l'exemple suivant présenté dans la figure 4.3. Le scénario 1 suivant doit être exécuté pour diffuser les valeurs P_0^i :

M1 : A → {B, C, D}
ID_A, Pk_A, $Cert_A$.

M2 : B → A
ID_B, ID_A, Pk_B, $Cert_B$.

M3 : A → {B, C, D}
(P_0^A) Sk_A.

M4 : B → A
(P_0^B) Sk_B.

ID_A : est l'identité du nœud A.
ID_B : est l'identité du nœud B.
P_0^A : est la valeur finale de la chaîne de hachage de A.
P_0^B : est la valeur finale de la chaîne de hachage de B.

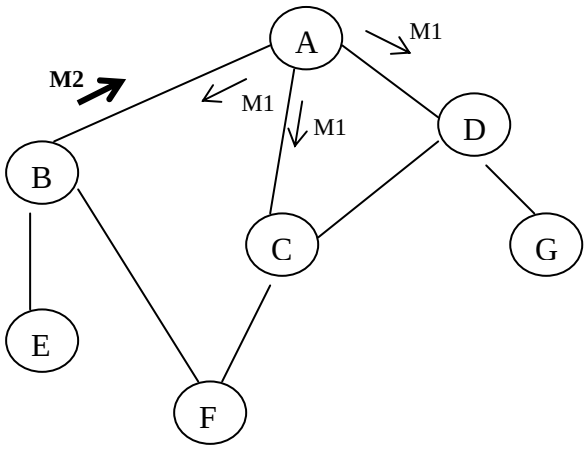

Fig. 4.3 : Diffusion des valeurs P_o.

Le nœud A diffuse d'abord sa clé publique et son certificat numérique à tous les nœuds voisins d'un seul saut. Dans cet exemple, le nœud A envoie le message M1 qui contient sa clé publique et son certificat numérique aux nœuds voisins B, C, D. Chacun de ses nœuds voisins vérifient la clé publique du nœud A, en utilisant le certificat qui est associé. De leurs part, les nœuds B, C, D envoient leurs clés publiques et leurs certificats numériques au nœud A et à tous leurs nœuds voisins d'un seul saut. Les deux messages M1 et M2 permettent l'échange des clés publiques qui est assuré une seule fois entre chaque paire de nœuds à l'initialisation du réseau. Quand le nœud A vérifie les clés publiques de ses nœuds voisins, il leurs envoie dans le message M3 sa valeur finale de hachage P_0^A, chiffrée avec sa clé privée Sk_A. De cette manière, ses nœuds voisins sont sûrs que seulement le nœud A qui possède la clé privée Sk_A a envoyé la valeur P_0^A. De la même manière, quand les autres nœuds vérifient les clés publiques de leurs voisins d'un seul saut, ils leurs envoient leurs valeurs finales de hachage.

A la fin de ce scénario, tous les nœuds qui forment le réseau ad hoc auront les valeurs finales de tous leurs nœuds voisins d'un seul saut. Chaque nœud sauvegarde les valeurs finales de hachage de ses voisins d'un seul saut dans une table appelée table de hachage, par exemple au niveau du nœud A on trouve la table suivante :

ID	**P_0**	……
B	P_0^B	…
C	P_0^C	…
D	P_0^D	…

Table 4.1 : La table de hachage au niveau du nœud A

4.3.3.2 Phase d'authentification

Quand le nœud A veut communiquer avec un nœud $_i$, il cherche d'abord la valeur finale de hachage de ce nœud $_i$ dans sa table de hachage. Dans le cas où il trouve la valeur P_0^i du nœud $_i$, le scénario 2 suivant sera exécuté :

M1 : A → i
Mess1, P_j^A, TIMES

M2 : i → A
Mess2, P_k^i, TIMES

Mess1 : le message que le nœud A veut envoyer au nœud i.
TIMES : permet d'éviter le rejoue de messages.

Le nœud A associe avec le message à envoyer Mess1, une valeur P_j^A ($0<j<N$) de sa chaîne de hachage, qui n'est pas déjà utilisée. Le nœud $_i$ applique une fonction

de hachage *j* fois sur la valeur P_j^A pour vérifier l'identité du nœud A. S'il trouve la valeur finale P_0^A déjà sauvegardée, il sera sûr que seulement le nœud A qui a envoyé le message M1. De la même manière, le nœud A sera sûr de l'identité du nœud $_i$ en appliquent *k* fois la fonction de hachage sur la valeur P_k^i.

Dans le cas où le nœud A veut envoyer un message à un nœud $_i$ dont la valeur finale de hachage P_0^i n'apparaît pas dans sa table de hachage. Dans ce cas, le nœud A envoie une requête de hachage REQ_hash qui contient l'identité du nœud $_i$ à ses voisins d'un seul saut dont il possède les valeurs finales de hachage, afin de récupérer la valeur finale de hachage P_0^i du nœud $_i$. De leurs part, ces nœuds envoient des requêtes de hachage à leurs nœuds voisins jusqu'à trouver le nœud $_i$ et récupérer sa valeur finale de hachage P_0^i. Le nœud qui possède la valeur finale de hachage du nœud $_i$ envoie une réponse REP_hash qui contient la valeur P_0^i. Les messages de requête REQ_hash et les messages de réponse REP_hash doivent être authentifiés par les nœuds en associant à chaque fois une valeur de hachage qui permet de s'assurer de l'identité de l'émetteur.

Dans l'exemple suivant de la figure 4.4, le nœud A veut communiquer avec le nœud G. Mais le nœud A ne trouve pas la valeur de hachage P_0^G dans sa table de hachage, donc il envoie une requête de hachage aux nœuds B, C, D. Le nœud B qui possède la valeur finale de hachage P_0^G envoie une réponse REP_hash au nœud A. Cette réponse contient la valeur finale de hachage de G associée avec une des valeurs de hachage du nœud B : P_k^B. La valeur P_k^B permet d'assurer au nœud A que la valeur P_0^G n'est pas envoyée par un attaquant. Donc les nœuds intermédiaires authentifient les messages qu'ils reçoivent avant de les envoyer.

Dans l'exemple suivant, si le nœud A veut envoyer un message au nœud G, il exécute le scénario 3 suivant :

REQ_hash : A → B, C, D
ID_A,ID_G, P_l^A, TIMES

REP_hash : B → A
P_0^G, P_k^B, TIMES

ID_G : l'identité du nœud cherché
P_0^G : la valeur finale du nœud G.

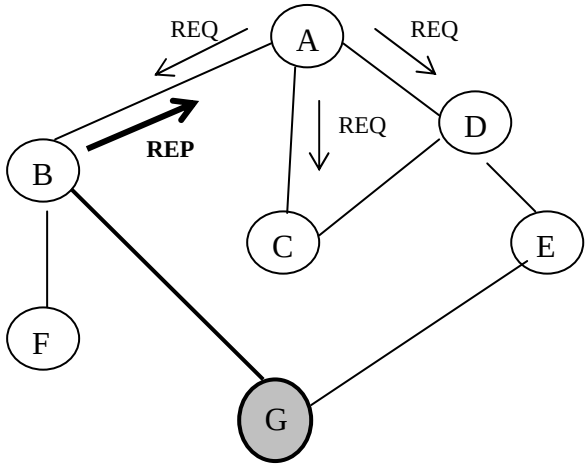

Fig. 4.4 : l'envoi d'une requête de hachage pour récupérer les valeurs P_0

Une entrée pour G est créée dans la table de hachage du nœud A et la valeur P_0^G est sauvegardée, voir table 4.2.

ID	**P_0**	……
B	P_0^B	…
C	P_0^C	…
D	P_0^D	…
G	**P_0^G**	…
…	…	…

Table 4.2 : Mise à jour de la table de hachage du nœud A.

4.3.4 Entretien du réseau

Dans cette partie, on décrit les étapes à exécuter dans le cas où un nouveau nœud veut joindre le réseau ou dans le cas où un nœud veut créer une nouvelle chaîne de hachage.

4.3.4.1 Nouveaux nœuds

Lorsqu'un nouveau nœud veut joindre le réseau ad hoc, il récupère d'abord une paire de clés publique / privée ainsi que le certificat numérique associée auprès de l'autorité de certification qui est off line. Il crée une chaîne de hachage de taille N. Ensuite il exécute le scénario 1 (deuxième étape : diffusion P_0); il envoie sa clé publique et son certificat numérique à tous ses voisins d'un seul saut. Par la suite, il leurs envoie sa valeur finale de la chaîne de hachage et il récupère leurs valeurs finales de hachage. Ensuite, il sauvegarde ses valeurs dans sa table de hachage. De cette manière il devient membre du réseau ad hoc, et il peut envoyer et recevoir des messages tout en assurant l'authentification.

Quand un nœud veut quitter le réseau, aucun problème ne se présente. Puisque c'est le seul qui connaît sa valeur finale de hachage et un attaquent ne peut pas prendre son identité.

4.3.4.2 Mise à jour

Quand un nœud ¡ utilise toutes ses valeurs de la chaîne de hachage[1], il crée une nouvelle chaîne de hachage. Il envoie ensuite la nouvelle valeur finale $P_{0\text{-}nv}^B$ à tous ses nœuds voisins qui mettent à jour leurs tables de hachage. La nouvelle valeur finale de hachage doit être associée avec la dernière valeur de la chaîne précédente qui n'est pas déjà utilisée pour assurer l'authentification. Par exemple (figure 4.5), si le nœud B utilise toutes les valeurs de sa chaîne de hachage P_1^B, P_2^B, P_3^B, ..., P_{N-1}^B sauf la valeur P_N^B, il génère une nouvelle valeur $P_{N\text{-}nv}^B$ et applique une fonction de hachage pour créer la nouvelle chaîne de hachage $P_{0\text{-}nv}^B$, $P_{1\text{-}nv}^B$, $P_{2\text{-}nv}^B$, ..., $P_{N\text{-}nv}^B$. Pour que les voisins du nœud B mettent à jour leurs tables de hachage avec la nouvelle valeur $P_{0\text{-}nv}^B$, le nœud B doit leurs envoyer le message de mise à jour MAJ_hash suivant :

MAJ_hash : B → A, G, F
ID_B, $P_{0\text{-}nv}^B$, P_N^B, TIMES.

[1] Un nœud ne doit pas utiliser toutes les valeurs de sa chaîne de hachage, il doit laisser au moins une valeur pour pouvoir mettre à jour la chaîne de hachage.

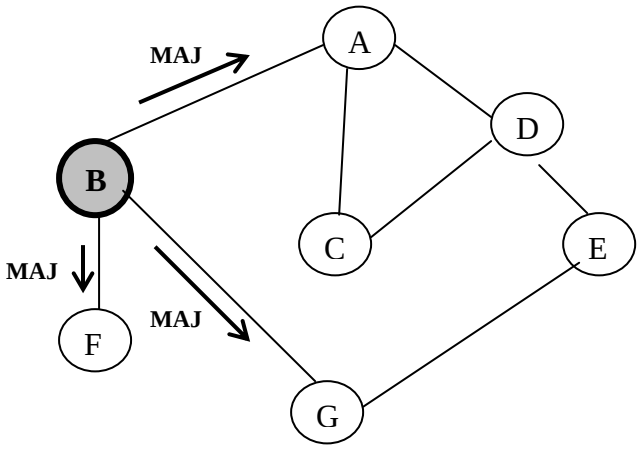

Fig. 4.5 : la mise à jour de la chaîne de hachage

La valeur P_N^B qui est envoyée avec la nouvelle valeur créée permet d'assurer aux nœuds A, G, F que seulement le nœud B qui a envoyé le message de mise à jour MAJ_hash.

4.3.5 Remarques

- ✓ Pour les réseaux ad hoc de petite taille, le nœud ¡ peut diffuser sa valeur P_0^i à tous les nœuds du réseau et pas seulement aux nœuds voisins. Dans ce cas, on n'aura pas besoin d'envoyer des requêtes de hachage pour récupérer les valeurs finales des autres nœuds, mais le problème est qu'il faut sauvegarder un tableau de hachage qui est volumineux puisqu'il contient toutes les valeurs finales de hachage et aussi le problème de la mise à jour de cet table de hachage.
- ✓ Notre protocole assure une authentification mutuelle puisque les deux nœuds qui veulent se communiquer s'échangent d'abord leurs valeurs finales de hachage pour vérifier leurs identités.
- ✓ Les valeurs P_0 sont de petites tailles, donc la mise à jour et la sauvegarde de ses valeurs ne présente pas un problème.

4.3.6 Protocole proposé

On décrit d'abord le format de différents paquets utilisés dans notre protocole, ensuite on décrit les étapes de l'algorithme.

4.3.6.1 Format des paquets

Dans cette partie, on décrit les formats généraux des paquets utilisés. Nous détaillons les champs utilisés dans l'algorithme présenté par la suite.

1. Paquet de données

Header	Data

- *Header* est l'entête du paquet et *Data* est la partie qui contient les données. Le champ *Header* contient les entêtes des différents protocoles utilisées. Il contient principalement :
- *Source* : l'adresse de l'émetteur (nœud source).
- *Destination* : la destination du paquet.
- *IdPacket* : un numéro de séquence géré par le nœud source.

2. Paquet de requête : REQ_hash

Source	Destination	IDENT	VERIF	IdPacket

Ce paquet est envoyé par un nœud dans le cas où il ne trouve pas dans sa table de hachage la valeur de hachage du nœud avec qui il veut se communiquer.

- *Source* : est l'adresse de la source de requête, c'est-à-dire du nœud qui cherche la valeur de hachage.
- Destination : est l'adresse du nœud voisin.
- *IDENT* : est l'identité du nœud avec qui le nœud source veut communiquer.
- *VERIF* : une valeur de hachage qui permet de s'assurer de l'identité de l'émetteur d'un paquet.

3. Paquet de réponse : REP_hash

Source	Destination	VERIF	HASH	IdPacket

Ce paquet de réponse est envoyé par un nœud qui reçoit le paquet de requête de hachage et qui possède la valeur recherchée.

- *Source* : est l'adresse de la source de la requête pour laquelle ce paquet de réponse est envoyé. C'est la destination de ce paquet de réponse.
- *Destination* : est l'adresse de l'émetteur du paquet.
- *VERIF* : une valeur de hachage qui permet de s'assurer de l'identité de l'émetteur du paquet.
- *HASH* : est la valeur de hachage recherché par le nœud Source.

4. Paquet d'erreur

Source	Destination	HASH	IdPacket

Ce paquet est envoyé quand une valeur de hachage n'est pas trouvé.

- *HASH* : est la valeur de hachage recherchée et non trouvée.

5. Paquet de mise à jour : MAJ_hash

Source	Destination	VERIF	NV_HASH	IdPacket

Ce paquet est envoyé quand un nœud utilise toutes ses valeurs de hachage et veut créer une nouvelle chaîne de hachage.

- *Source* : est l'adresse de la source de requête, c'est-à-dire du nœud qui a renouvelé sa chaîne de hachage.
- *Destination* : est l'adresse du nœud voisin.
- *NV_HASH* : est la nouvelle valeur de hachage créée par le nœud source.

4.3.6.2 Algorithme

Avant de déterminer l'algorithme de notre système, il est nécessaire de définir la terminologie à utiliser :

P_j^i : la j - ième valeur de la chaîne de hachage du nœud $_i$.
j : entier (0<j<N) indique la fin de la chaîne de hachage.
N : entier qui indique la taille de la chaîne de hachage.
bool : un booléen qui prend soit la valeur vrai ou faux pour indiquer si deux chaînes de caractères sont égaux ou non.
Pk_i : la clé publique du nœud $_i$.
Sk_i : la clé secrète du nœud $_i$.
$Cert_i$: le certificat numérique du nœud $_i$.
{voisoin_one_hop} $_i$: ensemble qui contient les voisins d'un seul saut du $nœud_i$.
{table_hachage} $_i$: ensemble qui contient les valeurs finales de hachage des nœuds voisins du nœud $_i$.
REQ_hash $_i$: requête de recherche d'une valeur finale de hachage du nœud $_i$.
REP_hash $_i$: la réponse à la requête de hachage, et qui contient la valeur de hachage recherché par le nœud $_i$.
MAJ_hash $_i$: contient la nouvelle valeur créée par le nœud $_i$.

Phase d'initialisation :

Etape1 : L'AC initialise les nœuds avec les paires de clés et les certificats numériques.

L'autorité de certification
{
Générer une paire de clés en utilisant l'algorithme asymétrique RSA pour chaque nœud $_i$;

Certif = **créer_certif** (Sk_{CA}, Pk_i) ;
Envoyer (Pk_i, Sk_i) au nœud $_i$;
Envoyer ($cert_i$) au nœud $_i$;

}

Chaque nœud $_i$ exécute l'algorithme distribué suivant :

Nœud $_i$
{
Recevoir (Pk_i, Sk_i) auprès de l'AC;
Recevoir ($cert_i$);
Générer aléatoirement une valeur P_N^i;
Sauvegarder (P_N^i);

Calculer P_{N-1}^i = **hash** (P_N^i);
Calculer P_{N-2}^i = **hash** (P_{N-1}^i);
...
…
Calculer P_0^i = **hash** (P_1^i);
}

/* Génération de la chaîne de
/ * hachage : P_0^i, …, P_{N-1}^i, P_N^i.

Etape2 : diffusion des valeurs P_0^i

Nœud $_i$:

Res_Chif=Chiffrer (P_0^i, Sk_i) ;
Tant que ∃ un voisin $_j$ ∈{voisoin_one_hop} $_i$
Faire
{
Envoyer (Pk_i, $Cert_i$) au voisin $_j$.
Quand le nœud $_i$ **reçoit** (Pk_j, $cert_j$);
Si verif_cerificat (Pk_{ca}, Pk_j)= vrai
Alors
{
Recevoir (P_0^j) ;
Sauvegarder (P_0^j) dans sa table de hachage ;
Envoyer (Res_Chif) au voisin $_j$;
Voisin $_j$ =voisin_suivant
}
Sinon ERREUR

} **Fait**

Phase d'authentification :

Le nœud $_i$ veut communiquer avec le nœud $_l$:

DEBUT
j=1 ;
// Chercher dans sa table de hachage la valeur P_0^l
Si $P_0^l \in$ table_ hachage $_i$
Alors
{ / * La valeur P_0^l est trouvée dans la table_hachage $_i$ */
Si (j < N)
Alors
{
Envoyer (P_j^i) avec le message à émettre (0<j<N) ;
j ++ ;
} fsi ;

Si (j>N) /* Toutes les valeurs de la chaîne de hachage sont utilisées
/* Donc il faut générer une nouvelle chaîne de hachage
Alors {
Générer une nouvelle valeur aléatoire x= $P_{N\text{-}nv}^i$
Générer une autre chaîne de hachage : $P_{0\text{-}nv}^i$, $P_{1\text{-}nv}^i$, …, $P_{N\text{-}nv}^i$;
/* Envoyer la nouvelle valeur de hachage à ses voisins */
Tant que ∃ un voisin $_j \in$ {voisoin_one_hop} $_i$
Faire
{Envoyer MAJ_hash $_i$ au voisin $_j$;**} fait ;**

} fsi ;

Sinon { /* La valeur P_0^l n'est pas trouvée dans la table
/* Envoyer une requête de recherche ;
Envoyer REQ_HASH $_l$ au voisin $_k \in$ {voisin_one_hop} $_i$;
etiq : pour tout voisin $_k$
Si $P_0^l \in$ {table_hachage}$_k$
Alors voisin $_k$ envoie REP_hash $_i$ au nœud $_i$
Sinon
{
Voisin $_k$ envoie REQ_hash $_l$ à {voisin_one_hop} $_k$;
Go to etiq
} fsi ;
}

} fsi ;

FIN

Les procédures utilisées dans cet algorithme sont décrites :

```
procedure hash(Mess)
  {
   /* Utiliser l'algorithme de hachage SHA-1 pour créer un condensat  de taille
fixe*/
  }
```

```
procedure chiffrer(Mess, Ski)
  {
  /* Utiliser la clé privée Ski et l'algorithme asymétrique RSA pour déchiffrer le
       message Mess */
  }
```

```
procedure déchiffrer(Mess, Pki)
  {
   /* Utiliser la clé publique Pki et l'algorithme asymétrique RSA pour chiffrer le
       message Mess */
  }
```

```
procedure créer_ certificat (SkCA, Pki)
  {
    h_pk = hash(Pki) ;
    /* Signer le condensat de la clé publique h_pk  en utilisant la clé secrète de
       l'AC*/
    signature = chiffrer (h_pk, SkCA) ;
    /* La valeur signature avec d'autres informations d'identité du nœud i
         constituent le certificat  numérique*/
   }
```

```
procedure vérifier _certificat (PkCA, val)
  {
   res = déchiffrer (val, PkCA);
   h_pk = hash(Pki) ;
    bool = comparer(h_pk, res) ;
    return (bool);
   }
```

4.3.7 Les avantages de cette solution

- ✓ Dans une solution basée sur CA, l'utilisateur doit propager sa clé publique et sa signature numérique (son certificat numérique). Dans notre système, les utilisateurs propagent leurs clés publiques aux voisins d'un seul saut seulement à l'initialisation du réseau, ensuite ils envoient les valeurs P_0.
- ✓ L'utilisation des fonctions de hachage et l'utilisation minimale des clés cryptographiques permettent à la solution d'être plus efficace et plus rapide.
- ✓ Pour n entités, une PKI nécessite n pairs de clés tandis qu'une SKI nécessite jusqu'à n (n-1)/2 clés partagées si chaque entité veut communiquer avec une autre entité. Dans notre schéma, un nœud $_i$ doit sauvegarder k < = n valeurs de chaînes de hachage (les valeurs de hachage de ses voisins d'un seul saut et des nœuds avec qui il a déjà communiqué).

4.3.8 Inconvénients

- ✓ Cette solution présente l'inconvénient de la mise à jour de la chaîne de hachage (des valeurs de la chaîne de hachage). A chaque fois qu'un nœud utilise toutes ses valeurs, il crée une autre chaîne, et il diffuse sa valeur à tous ses nœuds voisins.
- ✓ La signature d'une clé publique peut être vérifiée par plusieurs entités tandis qu'une valeur de la chaîne de hachage permet seulement une seule authentification mutuelle, qui est représenté par l'échange des valeurs de hachage.

4.4 Conclusion

Dans ce chapitre, nous avons proposé une solution qui traite le problème de l'authentification des nœuds dans les réseaux ad hoc. Nous avons exploité une caractéristique importante des fonctions de hachage qui est la minimisation de puissance de calcul et de capacité de stockage. Cette caractéristique est très appropriée aux réseaux ad hoc.

Puisque les clés publiques et les certificats numériques sont utilisés seulement dans l'étape de l'initialisation, le nombre d'opérations de clés publiques est réduit (les opérations de mise à jour de certificats numériques et de renouvellement des paires de clés).

Chapitre 5

Etude de performance des algorithmes cryptographiques utilisés dans notre protocole

Chapitre 5 : Etude de performance des algorithmes cryptographiques utilisés dans notre protocole

Partie 1 : Comparaison de performances des algorithmes cryptographiques

5.1 Introduction

Comme nous avons présenté précédemment, les unités mobiles dans les réseaux ad hoc se caractérisent par une bande passante modeste, des sources d'énergies limitées comme les batteries, vitesse de CPU et capacité de mémoire limitées.

D'après ces caractéristiques, il est utile d'étudier les performances de chaque algorithme cryptographique afin de justifier notre choix de l'utilisation des fonctions de hachage au lieu des algorithmes symétriques ou asymétriques.

Dans ce chapitre, nous allons présenter quelques travaux qui ont étudié les performances de différents algorithmes cryptographiques dans la première partie, ensuite nous allons étudier la performance de notre protocole dans la deuxième partie.

5.2 Consommation d'énergie des protocoles de sécurité

5.2.1 Introduction

Dans ce travail **[POT 03]**, les auteurs ont étudié les besoins de la consommation d'énergie du protocole de sécurité fréquemment utilisé : le protocole SSL (Secure Socket Layer). Pour faire une analyse d'énergie, les auteurs de cet étude ont exécuté des transactions de données sur un PDA iPAQ Compaq, calculant l'énergie consommée pendant les intervalles de temps dans lesquels le protocole de sécurité ou les algorithmes cryptographiques qui le constitue sont exécutés.

On présente dans ce qui suit les résultats de l'analyse faite sur la consommation d'énergie des algorithmes cryptographiques symétriques, asymétriques et des fonctions de hachage.

5.2.2 L'énergie consommée par les algorithmes cryptographiques symétriques

Les algorithmes symétriques peuvent être choisis parmi deux classes d'algorithmes pour être utilisé dans un protocole de sécurité : les algorithmes de chiffrement continu (qui convertissent le texte clair en texte chiffré un bit à la fois, par exemple : l'algorithme RC4) et la deuxième classe d'algorithmes est le chiffrement par bloc (ces algorithmes chiffrent des blocs de donnée de même taille par exemple : DES, 3DES, AES). Avant de commencer l'opération de chiffrement ou de déchiffrement pour les deux classes d'algorithmes, une clé d'entrée (généralement de taille 64 bits) est étendue afin de dériver une clé distincte pour chacune des rondes qui constituent l'algorithme (cette étape est appelée initialisation de la clé). Le chiffrement ou le déchiffrement dans les algorithmes symétriques procède ensuite par une séquence répétée (appelée ronde) des calculs mathématiques. Donc un algorithme symétrique est constitué de deux phases : l'initialisation de la clé et le chiffrement ou le déchiffrement.

La figure 5.1 suivante montre des variations dans la consommation d'énergie due à l'utilisation de différents algorithmes symétriques. Pour chaque algorithme

symétrique, il est décrit l'énergie nécessaire pour la phase d'initialisation de la clé et l'énergie nécessaire pour la phase de chiffrement ou de déchiffrement. Nous constatons d'après la figure que AES a le moindre coût d'énergie et que BLOWFISH a le plus grand coût.

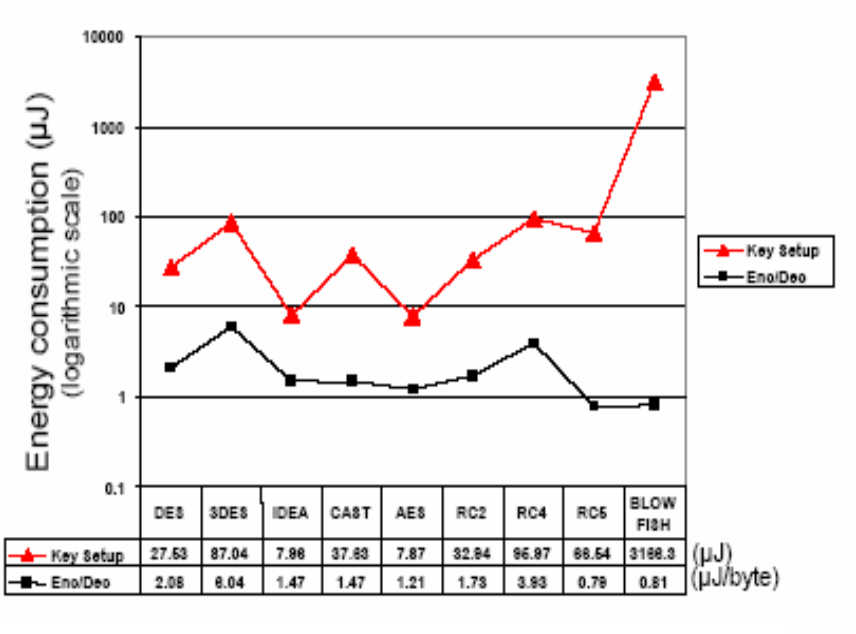

Fig. 5.1 : La consommation d'énergie des différents algorithmes asymétriques

Le coût élevé de BLOWFISH est principalement dû à son coût très élevé de l'initialisation des clés, puisque la clé étendue de BLOWFISH se compose des clés secondaires de total 4168 bits, ce qui lui fournit une sécurité très robuste. Le coût de chiffrement et de déchiffrement de BLOWFISH est tout à fait petit. Donc dans le cas des transactions de données très élevé, le coût de l'initialisation de la clé est diminué par le coût bas du chiffrement. Il est intéressant de noter que le coût d'énergie de IDEA, pour le chiffrement ou le déchiffrement et l'initialisation de la clé est presque similaire au coût de l'algorithme AES. Cependant, la résistance aux attaques de AES est supérieure à celle de l'IDEA, ce qui rend le premier algorithme plus intéressant.

5.2.3 Les fonctions de hachage

La table 5.1 suivante récapitule le coût d'énergie des algorithmes de hachage généralement utilisés. En général, les algorithmes de hachage sont moins complexes que les autres algorithmes cryptographiques, et devraient intuitivement consommer le moindre coût d'énergie. D'après la table suivante, on observe que les algorithmes MD2 et HMAC ont les plus grandes valeurs de consommation d'énergie par rapport aux autres algorithmes. SHA et SHA1 sont les plus nouveaux algorithmes de hachage, et ont plus de nombre d'étapes que MD4 et MD5. De plus, SHA et SHA-1 ont une meilleure résistance aux collisions (probabilité de deux entrées ayant la même valeur de hachage) que MD4 et MD5. Ces avantages de SHA et de SHA-1 les rendent plus favorisés par rapport aux algorithmes MD4 et MD5, bien que leur coût d'énergie est légèrement élevé. Ce qui explique notre choix de l'utilisation de l'algorithme SHA-1 dans notre protocole.

ALGORITHME	MD2	MD4	MD5	SHA	SHA-1	HMAC
ENERGIE (µJ/B)	4.12	0.52	0.59	0.75	0.76	1.16

Table 5.1 : Consommation d'énergie des fonctions de hachage

5.2.4 Les algorithmes asymétriques

La table 5.2 suivante compare la consommation d'énergie des trois algorithmes asymétriques qui sont approuvés par FIPS (Federal Information Processing Standard) pour la génération et la vérification des signatures. Ces trois algorithmes sont : RSA, DSA (digital signature algorithm) et ECDSA (elliptic curve digital signature algorithm). Notons que les auteurs ont utilisé dans leur analyse une clé de 163 bits pour les calculs dans l'algorithme ECDSA, qui est équivalent à une clé de 1024 pour RSA. Les valeurs de l'énergie sont liées aux trois étapes essentielles associées aux algorithmes de signature digitale : la génération de clé, la création de signature et la vérification de signature.

Algorithme	Taille clé (bits)	Génération de clé (mJ)	Signature (mJ)	Verification (mJ)
RSA	1024	270.13	546.5	15.79
DSA	1024	293.20	313.6	338.02
ECDSA	163	226.65	134.2	196.23

Table 5.2 : Le coût d'énergie des algorithmes asymétriques

Les résultats montrent que l'algorithme ECDSA consomme moins d'énergie que l'algorithme DSA. Cependant, les algorithmes RSA et ECDSA ont des coûts d'énergies complémentaires. RSA effectue la vérification de signature efficacement, alors qu'ECDSA impose un plus petit coût pour la génération de signature. La différence entre le coût d'énergie de la génération et de la vérification de signature dans l'algorithme RSA est plus grande que dans ECDSA. Si on veut exécuter de fréquentes générations de signature, donc il est préférable d'utiliser ECDSA. D'autre part, si la fréquence de la vérification de signature est plus grande que la génération de signature, donc l'algorithme RSA doit être utilisé.

5.2.5 Analyse de cette étude

D'après les résultats de cette étude, nous constatons clairement que les fonctions de hachage consomment moins d'énergie par rapport aux autres algorithmes cryptographiques symétriques ou asymétriques. Les fonctions de hachage MD4 et MD5 consomment le moindre coût d'énergie par rapport au autres algorithmes de hachage, mais les algorithmes SHA et SHA-1 ont une meilleure résistance aux attaques. Ce qui permet de favoriser ces algorithmes (SHA et SHA-1) malgré leur coût d'énergie légèrement élevé.

5.3 Nombre d'opérations cryptographiques exécutées par seconde

Dans l'article **[PEI 00]**, les auteurs ont présenté les résultats d'évaluation de différents algorithmes cryptographiques.

5.3.1 Introduction

Les algorithmes cryptographiques qui sont efficaces et simples à implémenter dans des microprocesseurs puissants comme pour les ordinateurs de bureau, ne peuvent pas être implémentés pour des microprocesseurs moins puissants des dispositifs sans fil. Les auteurs ont remarqué que quelques opérations cryptographiques qui prennent seulement quelques millisecondes ou moins et qui sont largement utilisés pour sécuriser les données et assurer l'authentification et l'intégrité dans les ordinateurs de bureau peuvent passer des secondes ou même des minutes pour s'exécuter sur des dispositifs sans fil. De plus, l'espace mémoire dans ces dispositifs est limité, ce qui introduit un autre problème pour l'implémentation de ces algorithmes pour les dispositifs de basse puissance.

5.3.2 Résultats de tests

Les mesures de performance sont exécutées en déterminant le nombre d'opérations exécutées dans une seconde pour chaque algorithme cryptographique. Les auteurs de cet article ont mesuré le nombre de blocs de données qui peuvent être chiffrées dans une seconde ainsi que le nombre de bits de données hachés par seconde.

La figure 5.2 montre le nombre de blocs de données de 64 octets qui peuvent être exécutées par seconde par chacun des algorithmes choisis. On appelle ce nombre, le nombre d'opérations par seconde que chaque algorithme est capable d'exécuter. On peut exécuter plus de 115.000 empreintes de MD5, 53.000 chiffrements de RC6, 23.000 chiffrements de DES, 220 vérifications de signature de RSA et 18 signatures de RSA par seconde. Une balance logarithmique est exigée pour illustrer de telles différences énormes.

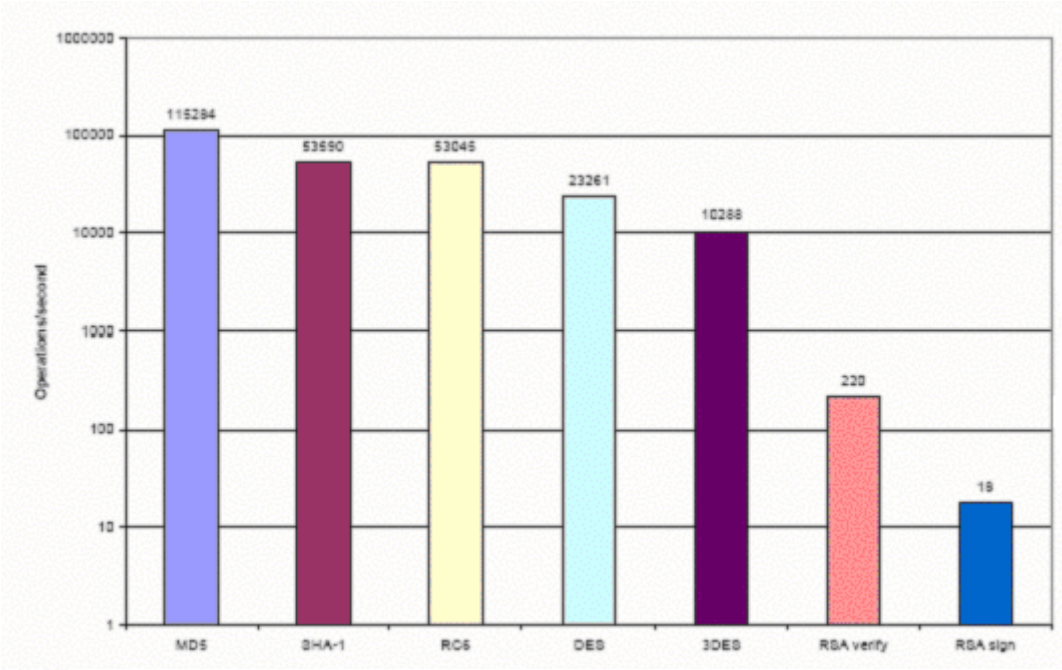

Fig. 5.2 Comparaison de différents algorithmes cryptographiques

5.3.3 Conclusion de l'analyse

Nous observons la différence dans les vitesses des fonctions de hachage, où MD5 est deux fois plus rapide que l'algorithme SHA-1. Cependant, SHA-1 est le plus

sécurisé, en raison des collisions ayant été trouvées dans la fonction de compression de l'algorithme MD5. La figure précédente montre aussi que la création ou la vérification d'une signature en utilisant l'algorithme RSA prend beaucoup plus de temps qu'un chiffrement avec un algorithme symétrique (RC6, DES, 3DES), ou plus encore de temps pour produire une empreinte digitale (SHA-1 ou MD5). Mais RSA est l'algorithme asymétrique le plus convenable pour exécuter des vérifications de signatures plus rapides, ce qui explique le choix des algorithmes dan notre protocole.

5.4 Temps d'exécution des algorithmes cryptographiques

Une autre étude **[ARG 01]** présente une analyse de performance de trois protocoles de sécurité les plus généralement utilisés dans les applications réseau. Le protocole SSL (Secure Sockets Layer), S/MIME et IPsec.

5.4.1 Paramètres des expériences

Pour faire leurs expériences, les auteurs ont utilisé HP (Compaq) iPAQ H3630 avec un processeur StrongARM 206 MHz et une RAM de 32MB, s'exécutant sur le système Windows CE Pocket PC 2002.

Toutes les expériences sont exécutées en utilisant des clés RSA de taille 1024 et 2048 bits avec un petit exposant publique (e=65,537) afin de rendre les opérations de clé publique plus rapide que les applications de clé privée. Les clés de taille 512 bits sont trop courtes pour des données sensibles et ne peuvent pas donc être employés dans les expériences qui essayent de satisfaire les besoins réalistes des transactions sécurisées.

5.4.2 Résultats de tests

Dans cette étude, on présente quelques mesures de temps concernant les algorithmes cryptographiques utilisées dans les trois protocoles étudiés. Le résultat de ces tests est présenté dans la table 5.3 suivante :

OPERATION	**TEMPS**	**ITERATIONS**
DES	7.354 secondes (7,354 ms)	100,000 chiffrements et 100,000 déchiffrements
SHA	19,111 secondes (19,111 ms)	100,000
RSA 1024 bits signature	782.125 secondes (782,125 ms)	10,000
RSA 1024 bits vérification	50.125 secondes (50,125 ms)	10,000
RSA 2048 bits signature	4,972.798 secondes (4,972, 798 ms)	10,000
RSA 2048 bits vérification	156.006 secondes (156,006 ms)	10,000

Table 5.3 : Les mesures de temps des primitives cryptographiques utilisées sur un iPAQ H3630.

5.4.3 Conclusion de cette étude

D'après cette étude, nous constatons que les opérations des algorithmes à clés publiques et précisément la signature et la vérification de signature avec l'algorithme RSA sont très lentes par rapport à la création d'un condensat avec l'algorithme de hachage SHA.

Partie2 : Etude de performance du protocole proposé

5.5 Introduction

Le choix des algorithmes cryptographiques utilisés dans notre protocole dépend essentiellement de la vitesse d'exécution de ces algorithmes. L'espace de stockage et la taille des messages dépendent de la taille des clés de ces algorithmes, de la taille de l'empreinte de hachage, ainsi que des tailles de signatures et de certificats. Tous ces facteurs dépendent des algorithmes utilisés.

5.6 Le choix des algorithmes cryptographiques

Après l'étude de performances des algorithmes cryptographiques et la comparaison de ces algorithmes dans la première partie de ce chapitre, nous avons choisi (table 5.4) d'utiliser dans notre protocole la fonction de hachage à sens unique SHA-1 qui permet de créer des valeurs de hachage afin de constituer une chaîne de hachage. Nous avons choisi aussi d'utiliser l'algorithme asymétrique RSA dans l'étape d'initialisation. L'algorithme RSA est utilisé pour générer des paires de clés publiques / privées et des certificats numériques. Ces clés permettent d'échanger les valeurs de hachages entre les nœuds du réseau tout en assurant l'authentification.

La fonction	Algorithme	Taille clé (bits)
Le hachage	SHA-1	-
La signature	RSA	1024

Table 5.4 : Algorithmes cryptographiques utilisés dans notre protocole

5.7 Capacité de stockage

Notre protocole nécessite un nombre de bits pour stocker les informations utiles afin d'assurer l'authentification des nœuds. Un espace de stockage est nécessaire pour sauvegarder les certificats numériques, les clés publiques et les clés privées utilisés.

On suppose qu'un champ de donnée moyen est de taille 128 bits, comme par exemple le champ identité (ID). La clé publique est composée d'un exposant publique e, de taille 16 bits, avec 1024 bits modulu n, et 32 bits pour une information de longueur comme spécifié par ISO9797 **[ISO91]**, donc la clé publique est de taille 16+1024+32= 1072 bits. Tandis que le certificat numérique est constitué de la clé publique du nœud (de taille 1072 bits), l'identité du nœud, l'identité de l'autorité de certification, et une date d'expiration du certificat, chacune de taille (128 bits), il est constitué aussi de la signature de la clé publique de taille 1024 bit. Donc la taille totale d'un certificat numérique est égale à 1072+ (128*3) +1024 = 2480 bits.

On détermine donc l'espace de stockage nécessaire dans la mémoire de chaque nœud. A l'initialisation du réseau, chaque nœud doit sauvegarder une paire de clés publique / privée qui est générée en utilisant l'algorithme à clé publique RSA. La longueur de clé est de 1024 bits. Le nœud doit sauvegarder aussi le certificat numérique associée à sa clé publique. Le certificat numérique contient la signature RSA ainsi que d'autres informations d'identité, sa taille est égale à 2480 bits (table 5.5). L'espace de stockage nécessaire est (1024 +1072+2480 = 4576 bits), c'est-à-

dire la taille des deux clés et du certificat numérique. Ensuite après que le nœud crée la chaîne de hachage, il sauvegarde aussi la valeur initiale générée de taille 160 bits (l'empreinte digitale de la fonction de hachage SHA-1).

Objet	**Taille (bits)**
Un champ moyen	128
Valeur de hachage SHA	160
Clé privée RSA (d)	1024
Signature RSA	1024
Clé publique RSA (e, n)	1072
Certificat clé publique RSA	2480

Table 5.5 : La taille des différents champs.

L'avantage de notre solution par rapport aux solutions basées sur les clés publiques est que dans notre protocole et après que le nœud utilise la paire de clés et le certificat numérique pour envoyer la valeur de la chaîne de hachage, l'espace mémoire nécessaire pour sauvegarder ces informations est libéré, il reste seulement l'espace mémoire réservé pour la valeur de hachage. Donc notre protocole permet de minimiser l'espace de stockage.

5.8 Temps d'exécution

Pour les solutions basées sur les clés publiques, le temps d'exécution inclut le temps de génération de signature et le temps de vérification de signature. Ainsi, pour assurer l'authentification, le nœud doit exécuter l'algorithme à clé publique RSA afin de vérifier la signature du nœud émetteur de message, ce qui nécessite un temps d'exécution égale à 5 millisecondes (d'après la table 5.3). Tandis que dans notre protocole, si un nœud veut s'assurer de l'identité du nœud émetteur, il doit vérifier la valeur de hachage reçue avec le message, c'est-à-dire il doit exécuter la fonction de hachage plusieurs fois pour trouver la valeur initiale du nœud émetteur du message. Au maximum, il doit exécuter la fonction de hachage 100 fois, ce qui représente la taille de la chaîne de hachage. Donc le temps nécessaire pour vérifier l'authentification est au maximum 0.02 * 100 = 2 millisecondes.

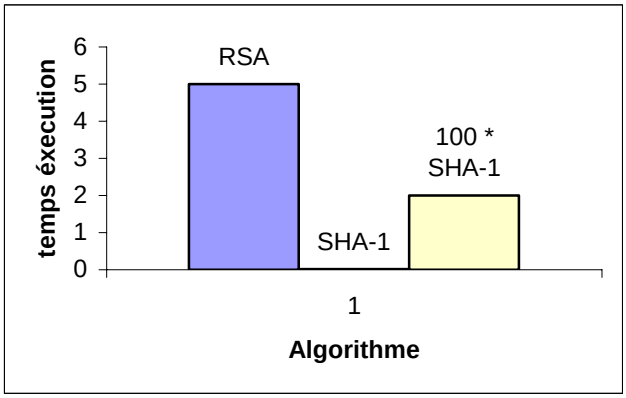

Fig. 5.3 Le temps d'exécution des algorithmes RSA, SHA-1.

5.9 La consommation d'énergie

L'énergie nécessaire pour exécuter une opération de vérification de signature en utilisant l'algorithme à clé publique RSA est égale à 15.79 mJ, tandis que l'énergie nécessaire pour produire une empreinte digitale ou pour vérifier cette empreinte avec l'algorithme SHA-1 est égale à 0.76 µJ. Ainsi pour garantir l'authentification, notre protocole consomme une énergie égale à 0.76 * 100 =76 µJ = 0.076 mJ, ce qui est très inférieur par rapport à l'énergie consommée par une seule vérification de signature (figure 5. 4).

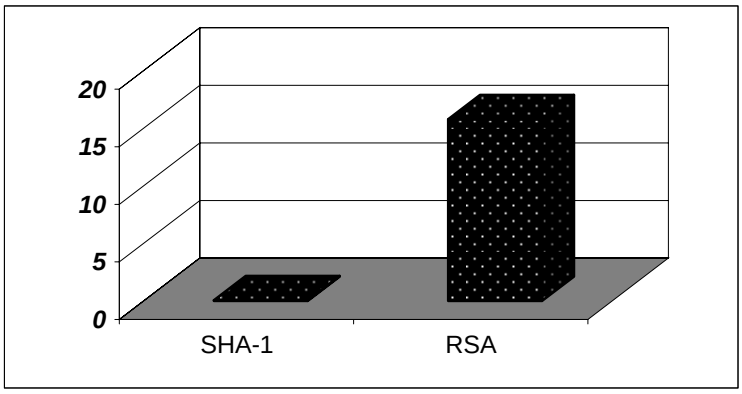

Fig. 5.4 La consommation d'énergie des algorithmes RSA, SHA-1.

5.10 Conclusion

Cette étude décrit une comparaison efficace de plusieurs algorithmes de hachage avec des algorithmes à clé symétrique et des algorithmes à clés publiques. Le premier résultat ou la première remarque est sur le nombre d'opérations effectuées par seconde. En général, on remarque que l'opération de hachage est plus rapide que l'opération de chiffrement symétrique, plus rapide que la vérification de signature ainsi que la génération de signature.

Notre protocole montre un gain important en consommation d'énergie et en temps d'exécution, il minimise aussi l'espace de stockage, ce qui correspond mieux aux caractéristiques limitées des réseaux ad hoc.

Conclusion générale

La sécurité des réseaux mobiles et spécialement des réseaux mobiles ad hoc n'a jamais cessé de susciter des préoccupations du fait qu'ils sont exposés à des menaces supplémentaires par rapport aux réseaux filaires. En général, ces menaces viennent du fait que les communications sans fil sont transmises par ondes radios et peuvent être écoutées par des personnes non autorisées.

Les techniques de chiffrement les plus utilisées dans les systèmes filaires ne sont pas toujours convenables pour les systèmes sans fils vu leurs caractéristiques limitées (puissance de calcul, capacité de stockage, bande passante).

L'utilisation des schémas cryptographiques nécessite généralement l'utilisation d'un service de gestion de clés. Pour cela, il existe des procédures basées sur une infrastructure à clé publique permettant la gestion de clés. Cette solution est à moitié satisfaisante car elle suppose l'existence d'une infrastructure centralisée alors que le concept de réseaux ad hoc appelle à une infrastructure distribuée. Par conséquent, les mécanismes de sécurité dans un réseau ad hoc devront être distribués.

Il est présenté dans ce travail différentes approches pour assurer l'authentification dans un réseau ad hoc. La plupart de ces solutions utilisent des certificats numériques ou des algorithmes de distribution de clés pour garantir l'authentification des nœuds. Ces solutions sont complexes et nécessitent une grande puissance de calcul.

Dans notre travail, nous avons exploité l'efficacité et la rapidité des fonctions de hachage pour proposer une solution au problème d'authentification dans les réseaux ad hoc. Nous avons utilisé les chaînes de hachage qui sont basées sur les fonctions de hachage à sens unique. Nous avons donc exploité deux caractéristiques importantes des fonctions de hachage. La première caractéristique est que la fonction de hachage est à sens unique, ce qui permet d'utiliser chaque valeur de hachage pour assurer l'authentification de son émetteur. La deuxième caractéristique est la rapidité de ses fonctions en les comparant avec les autres algorithmes cryptographiques (symétriques ou asymétriques), ainsi que la capacité de stockage minimale.

Le protocole que nous avons proposé montre un gain important en consommation d'énergie et en temps d'exécution, ce qui correspond mieux aux caractéristiques limitées des réseaux ad hoc.

Références

[ALB 01] : Patrick Albers, Olivier Camp, Jean-Marc Percher, Bernard Jouga, Ludovic Mé, Ricardo Puttini, **«Security in Ad Hoc Networks : a General Intrusion Detection Architecture Enhancing Trust Based Approaches»,** ESEO, Supélec, France, 2001.

[ALI 04] : Aliouane Lynda, Nadjib Badache **« User Authentication in Mobile Ad hoc Networks»**, publication in the International Conference on Pervasive Services, ACS/IEEE, 2004.

[ARC 00] : Jean-Luc Archimbaud, **«Certificats (électroniques) Pourquoi? Comment ?»**, CNRS/UREC, décembre 2000.

[ARG 01] : Patroklos G. Argyroudis, Raja Verma, Hitesh Tewari, Donal O'Mahony, **«Performance Analysis of Cryptographic Protocols on Handheld Devices»**, University of Dublin, Trinity College. 2001.

[ASO 00] : N. Asokan, P. Ginzborg, **«Key Agreement in Ad Hoc Networks»**, Computer Communications Volume 23, 2000.

[BAD 98] : Nadjib Badache, **« La mobilité dans les systèmes répartis »**, janvier 1998.

[BAG 95] : Aline Baggio, **«Environnements Mobiles : Etude et Synthèse Bibliographique»**, INRIA Rocquencourt, SOR Project, 1995.

[BAL 02] : D. Balfanz, D. K. Smetters, P. Stewart, H. Chi Wong, **«Talking To Strangers: Authentication in Ad-Hoc Wireless Networks »**, Internet Society, Conference Proceeding of NDSS Conference, 2002.

[BEC 98] : K. Becker, U. Willie, **«Communication complexity of group key distribution»**, Proceedings of 5th ACM Conference on Computer and communications Security, ACM Press, 1998.

[BER 00] : Berbar Ahmed, Saadi Rachid, **«Protection de la messagerie électronique par des méthodes cryptographiques»,** thèse d'ingénieurs, Université des sciences et de la technologie Houari Boumediene, USTHB, 2000.

[BHA 03] : Raghav Bhaskar, **« Group Key Agreement in Ad hoc Networks»**, INRIA Rocquencourt, rapport de recherche n° 4832, mai 2003.

[BID 95] : C. Bidon, V. Issamy : **« Un aperçu des problèmes de sécurité dans les systèmes informatiques »**, IRISA (Institut de Recherche en Informatique et Systèmes Aléatoires)- Publication interne N°959, octobre 1995, France.

[BIN 00] : Joseph Binder, Hans–Peter Bischof, Alan Kaminsky, **«Decentralized Key Management in Ad Hoc Networks»**, Rochester Institute of Technology, 2000.

[CAP 03] : Srdjan Capkun, Levente Buttyan, Jean-Pierre Hubaux, **«Self-Organized Public-Key Management for Mobile Ad Hoc Networks»**, Lausanne, Switzerland, 2003.

[COR 99] : S. Corson, J. Mackr , **« Mobile Ad hoc Networking (MANET) »**, RFC 2501, 1999.

[DAE 99] : J. Daemen, V. Rijmen. **AES Proposal : Rijndael**. AES Algorithm Submission, Sep 1999. http://www.nist.gov/aes.

[DER 01] : Derhab Abdelwahid, Djenouri Djamel, **«Simulation des algorithmes de routage dans les mobiles ad hoc »,** thèse d'ingénieurs, Université

des sciences et de la technologie Houari Boumedienne, USTHB, juin 2001.

[DJE 01] : Djenouri Djamel, Derhab Abdelwahid, **«Simulation des algorithmes de routage dans les mobiles ad hoc »,** thèse d'ingénieurs, Université des sciences et de la Technologie Houari Boumediene USTHB, juin 2001.

[EVA 74] : A. Evans, W. Kantrowitz, E. Weiss. **«A user authentication scheme not requiring secrecy in the computer»**. Communications of the ACM, 17(8):437-42, August 1974.

[FOK 02] : Klas Fokine, **«Key Management in Ad Hoc Networks»**, University Linkoping, 2002.

[FRO 00] : Magnus Frodigh, Per Johansson, Peter Larsson, **«Wireless ad hoc networking, the art of networking without a network»**, Ericsson Review No. 4, 2000, pp 248-263.

[GEN 96] : R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. **«Robust and efficient sharing of RSA functions»**, N. Koblitz, editor, Advances in Cryptology, Crypto '96 (Lecture Notes in Computer Science 1109), U.S.A, 1996.

[GOK 02] : Shardul Gokhale, Partha Dasgupta, **«Distributed Authentication for Peer-to-Peer Networks»**, Arizona State University, 2002.

[GRA 03] : Gilles Grandfils, Patrick Brosset, **« Borne d'Imprégnation d'Objet Communiquant »**, projet de fin d'étude, INSA de Lyon, France, 2003.

[HER 95] : A. Herzberg, S. Jarecki, H. Krawczyk, M. Yung, **«Proactive secret sharing or : How to cope with perpetual leakage»**, In D. Coppersmith, editor, Advances in Cryptology, Crypto '95 (Lecture Notes in Computer Science 963), California, U.S.A., August 1995.

[HIE 01] : M. Hietalahti, «**Key Establishment in Ad-Hoc Networks**», Laboratory for Theoretical Computer Science, Helsinki University of Technology, 2001.

[HIP 02] : Projet Hipercom, Communication à hautes performances, Rocquencourt, 2002, http://www.inria.fr

[HUB 01] : J-P. Hubaux, L. Buttyán, S. Capkun, **«The Quest for Security in Mobile Ad Hoc Networks»**, ACM 2001.

[HUB 03] : Srdjan Capkun, Jean-Pierre Hubaux, Levente Butty, **«Mobility Helps Security in Ad Hoc Networks»**, Switzerland.

[ISO91] : ISO/IEC 9796 : Information technology – security techniques – digital signature scheme giving message recovery, Juillet 1991.

[JAR 95] : S. Jarecki, **«Proactive secret sharing and public key cryptosystems»**, Master's thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, September 1995.

[JAR 96] : R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin. **«Robust threshold DSS signatures»**, U. Maurer, editor, Advances in Cryptography, Proceedings of Eurocrypt '96 (Lecture Notes in Computer Science 1070), Zaragoza, Spain, May 1996.

[KLE 00] : Matthieu Klein, **«PGP et applications cryptographiques»**, rapport de licence EEA sur les techniques de chiffrement, 2000.

[KON 01]: J. Kong, P. Zerfos, H. Luo, S. Lu, L. Zhang, **«Providing Robust and Ubiquitous Security»**, Support for Mobile Ad-Hoc Networks, IEEE ICNP, 2001.

[KRA 01] : Seung Yi, Robin Kravets, **«Practical PKI for Ad Hoc Wireless Networks»**, University of Illinois at Urbana-Champaign, USA, 2001.

[LAM 81] : L. Lamport, “Password Authentication with Insecure Communication”, communication of the ACM, vol. 24, no. 11, nov. 1981, pp. 770-72.

[LAW 02] : Y. W. Law, P. Hartel, S. Etalle, **« Security of Ad Hoc Networks : A Preliminary Discussion »**, September 2002.

[LEM 00] : Tayeb Lemlouma, **« Le routage dans les réseaux mobiles Ad Hoc »,** USTHB, octobre 2000.

[LUO 00] : H. Luo, S. Lu, **«Ubiquitous and Robust Authentication Services for Ad Hoc Wireless Networks»**, Technical Report 200030, UCLA Computer Science Department, 2000.

[LUO 02]: H. Luo, P. Zerfos, J. Kong, S. Lu, L. Zhang, **«Self-securing Ad Hoc Wireless Networks»**, IEEE ISCC, 2002.

[MAA 01] : Maarit Hietalahti, **«Efficient Key Agreement for Ad-hoc Networks»**, Master’s Thesis, Helsinki University of Technology, 2001.

[MAL 01] : Morteza Maleki, Karthik Dantu, Massoud Pedram, **«Power-aware Source Routing Protocol for Mobile Ad Hoc Networks»**, University of Southern California, Los Angeles, 2001.

[PEI 00] : Michael Peirce, **«Multi-Party Electronic Payments for Mobile Communications »**, A thesis submitted for the degree of Doctor of Philosophy in Computer Science, University of Dublin, 2000.

[POT 03] : Nachiketh R. Potlapally, Srivaths Ravi, Anand Raghunathan, Niraj K. Jha, **«Analyzing the energy consumption of security protocols»**, ISLPED’03, Seoul, Korea, August 2003.

[RIK 02] : Eric Ricardo Anton, Otto Carlos Muniz Bandeira Duarte, **«Group Key Establishment in Wireless Ad Hoc Networks»**, Universidade Federal do Rio de Janeiro, Brazil, 2002.

[ROG 98] : Phillip Rogaway, Don Coppersmith. **A software-optimized encryption algorithm**. Journal of Cryptology, 11(4), First Quarter 1998.

[ROY 99]: Elizabeth M. Royer, University of California, Santa Barbara, Chai-Keong Toh, Georgia Institute of Technology, **«A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks»**, 1999**.**

[SAL 02] : A. O. Salako, **«Authentication in Ad hoc Networking»** University College London, 2002.

[SCH 96] : Bruce Schneier, **«Applied Cryptography»**, Second edition, 1996.

[STA 99] : F. Stajano, R. Anderson, **«The Resurrecting Duckling : Security Issues in Ad-Hoc Wireless Networks»**, Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science, Springer-Verlag, 1999.

[SUN 01] : Jun-Zhao Sun, **«Mobile Ad Hoc Networking: An Essential Technology for Pervasive Computing»**, University of Oulu, Finland, 2001.

[VEN 02] : Lakshmi Venkatraman, Dharma P. Agrawal, **« A novel authentication scheme for ad hoc networks»**, Proceedings of IEEE Conference on Wireless Communications and Networking, vol.3, Chicago, USA, Sept. 2000.

[WEI 01] : André Weimerskirch, Gilles Thonet, **«A Distributed Light-Weight Authentication Model for Ad-hoc Networks»**, The 4th International Conference on Information Security and Cryptology (ICISC 2001), December 2001, Seoul, South Korea.

[WES 03] : A. Weimerskirch and D. Westhoff. **«Zero Common-Knowledge Authentication for Pervasive Networks»**, Tenth Annual International Workshop on Selected Areas in Cryptography (SAC 2003), 2003.

[YAN 01] : Zheng Yan, **«Security in Ad Hoc Networks»**, Networking Laboratory, Helsinki University of Technology, 2001.

[ZHA 00] : Muxiang Zhang, Christopher Carroll, Agnes H. Chan. **«The software-oriented stream cipher SSC2»**. Fast Software Encryption Workshop 2000, 2000.

[ZHO 99] : L. Zhou, Z. J. Haas, **«Securing Ad Hoc Networks»**, IEEE Networks, Volume 13, Issue 6, 1999.

[ZIM 95] : P. R. Zimmermann, **«The Official PGP User's Guide»**, MIT Press, Cambridge, Massachusetts, 1995.