L'Authentification dans les Réseaux Ad Hoc

ALIOUANE Lynda *, BADACHE ** Nadjib
** CERIST,*
*** LSI, USTHB, BP 32, EL-Alia, Bab-Ezzouar, 16111, Alger.*
E-Mail :{lyn_f35@yahoo.fr, badache@wissal.dz}

Introduction

Un réseau Ad Hoc est un réseau temporaire, formé d'un ensemble de nœuds qui peuvent communiquer entre eux sans aucune forme d'administration centrale. Un réseau ad hoc peut être utilisé dans les scénarios où il n'existe aucune infrastructure, ou l'existence d'infrastructure ne répond pas aux besoins d'applications telles que la sécurité ou le coût. Les applications militaires, les opérations de secours et les conférences sont des exemples où les réseaux ad hoc sont utilisés, mais la sécurité de la communication est nécessaire dans ces applications.

Dans un réseau ad hoc, les nœuds 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 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 **[1, 2]**.

Il existe plusieurs recherches récentes qui permettent d'assurer la sécurité dans les réseaux ad hoc. Beaucoup d'effort a été mis pour assurer la sécurité dans le but de distribuer les fonctionnalités du CA à un ensemble de nœuds dans le réseau **[1, 3]**. Une autre solution proposée dans **[4]** décrit un schéma d'authentification basé sur le système PGP dans le sens où les utilisateurs eux mêmes gèrent leurs certificats en se basant sur leurs connaissances personnelles. Une approche différente proposée par Asokan et Ginzboorg dans **[5]** est basée sur un mot de passe partagé. Dans cette solution, les nœuds voulant établir une session sécurisée doivent d'abord partager un secret qui est utilisé pour dériver un mot de passe fort. Un autre travail propose une solution basée sur le système Kerberos pour authentifier les utilisateurs mobiles dans les réseaux ad hoc **[6]**. La plupart de ces solutions sont basés sur la cryptographie asymétrique qui nécessite une puissance de calcul élevée.

Dans notre solution, nous avons exploité l'efficacité 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.

L'organisation de l'article est la suivante. Dans la section 2 de cet article, nous donnons un aperçu de différentes approches proposées dans la littérature ainsi qu'une analyse de chaque approche. Dans la section 3, nous fournissons le schéma de notre système et décrivons comment l'authentification est vérifiée en utilisant des chaînes de hachage. Enfin nous concluons l'article dans la section 4.

Les travaux de recherche

2.1 La cryptographie à seuil

Dans **[1],** les auteurs ont proposé d'utiliser la cryptographie à seuil (threshold cryptography) pour distribuer les fonctionnalités d'une autorité de certification parmi les nœuds du réseau. Dans l'article **[1],** la clé privée de l'autorité de certification qui sert à signer les clés publiques des nœuds est partagée parmi un ensemble de *n* nœuds du réseau.

Un schéma à seuil (*t*, *n*) permet de distribuer les services de l'autorité de certification à un groupe de n nœuds serveurs, de façon à ce que *t* nœuds puissent exécuter ensemble les services de l'autorité de certification, tandis qu'il est impossible d'exécuter ces services par moins de *t* nœuds **[7, 1]**.

Cette solution 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. D'autre part, cette solution 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* 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.

2.2 Les chaînes de certificats

Cette approche est proposée dans **[8]** et fournit une solution de gestion de clés publiques similaire à la solution PGP, dans le sens où les certificats sont délivrés par les utilisateurs eux-mêmes, en se basant sur leurs connaissances personnelles et sans l'implication d'aucune entité centralisée **[7]**. Dans ce schéma les nœuds sauvegardent les certificats délivrés pour les autres nœuds dans un répertoire local. Quand deux nœuds veulent s'authentifier, ils essayent de trouver une chaîne de certificat en combinant leurs répertoires locaux. Si une chaîne de certificat existe, le nœud est authentifié. Un mécanisme de révocation est absent dans ce schéma.

L'avantage principal de cette solution est qu'elle ne nécessite aucune forme d'infrastructure **[4]**. La gestion des clés et des certificats est effectuée par les utilisateurs eux mêmes.

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 où les nœuds n'ont aucune relation de confiance préalable **[4]**. 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 **[7]**.

2.3 Accord de clé dans un groupe (Key agreement)

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. 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 **[9]**. Par conséquent, chaque participant contribue une partie de la clé et signe cette donnée en utilisant le mot de passe faible.

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

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 utilisé pour échanger un secret dans un groupe, suppose que les nœuds participant au système sont arrangés dans une forme particulière, par exemple dans un hypercube.

Présentation d'un nouveau schéma d'authentification

3.1 Introduction

Comme nous l'avons dit dans la partie précédente, 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 fréquentes 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 la section précédente, pour assurer l'authentification dans un réseau ad hoc. 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 notre travail, 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. 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.

3.2 Les chaînes de hachages

3.2.1 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. Pour toute fonction de hachage, la probabilité d'une collision[1] est estimée à 2^n où n est la taille de l'empreinte.

[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).

Les auteurs de **[10]** 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 ».

3.2.2 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 :

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

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*.

3.2.3 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 **[11]**. 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 1).

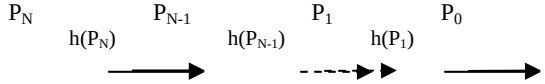

Figure 1 : Génération d'une chaîne de hachage

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 reçue. 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 **[12]** 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.

3.3 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[2]. 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).

3.3.1 Supposition

- Chaque nœud $_i$ a un identificateur unique ID_i.
- Chaque nœud possède une paire de clés publique / privée.
- Chaque nœud utilise une fonction de hachage à sens unique h (.). On utilise dans notre protocole la fonction cryptographique SHA-1.
- Chaque nœud a un certificat numérique signé par une AC digne de confiance
- Les liens de communications sont fiables et bidirectionnels.

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

- 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 **[12].**

3.3.2 Les étapes de l'authentification

Notre protocole **[13]** 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.

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 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 $Cert_i$. 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 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.

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 2. 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.

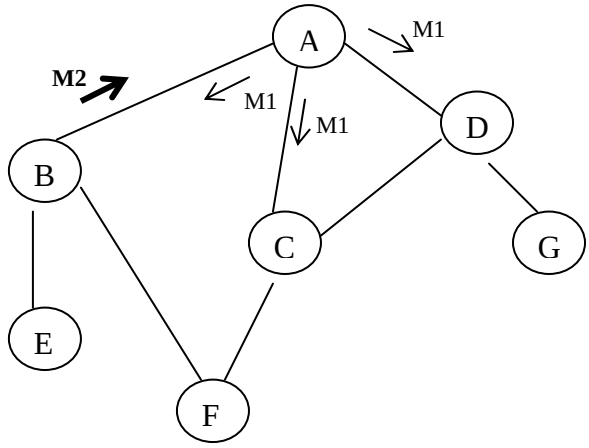

Figure 2 : Diffusion des valeurs P_o.

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 1 : La table de hachage au niveau du nœud A

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 3, 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.

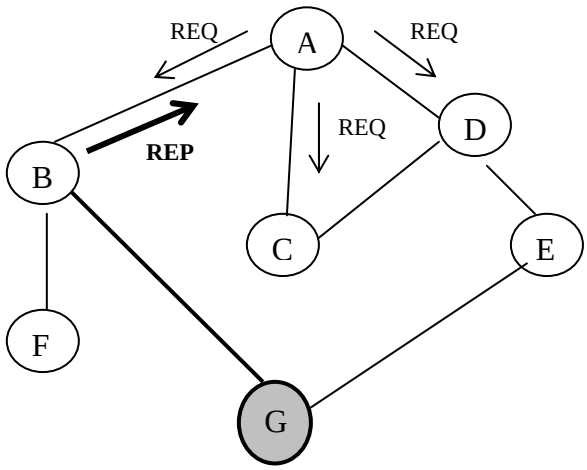

Figure 3 : L'envoi d'une requête de hachage pour récupérer les valeurs P_0

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

3.3.3 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.

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.

Mise à jour

Quand un nœud $_i$ utilise toutes ses valeurs de la chaîne de hachage[3], 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), 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 :

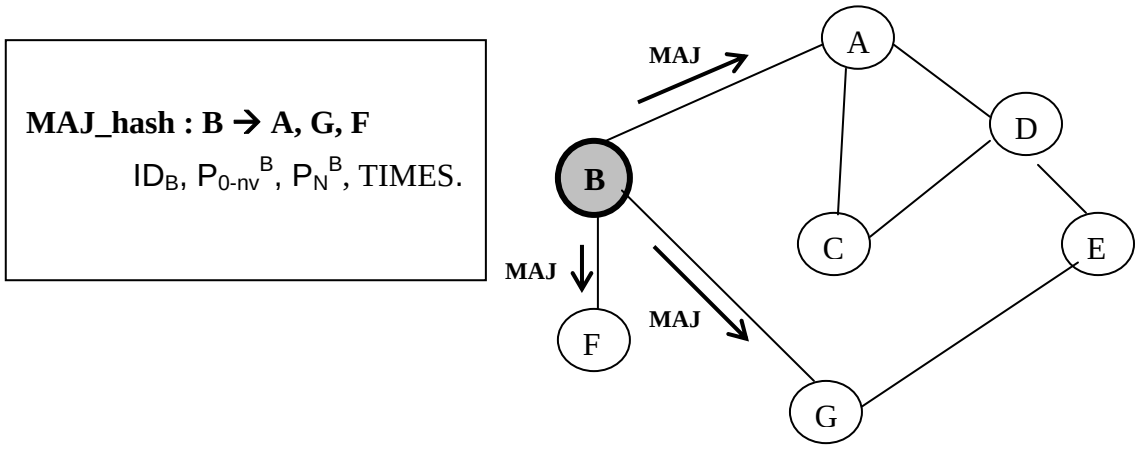

Figure 4 : la mise à jour de la chaîne de hachage

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.

3.3.4 Remarques

- ✓ Pour les réseaux ad hoc de petite taille, le nœud $_i$ 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.

[3] 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.

- ✓ 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.

3.3.5 Format des paquets du protocole proposé

On décrit le format de différents paquets utilisés dans notre protocole.

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 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.

3.3.6 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).

3.3.7 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 authentification mutuelle, qui est représenté par l'échange des valeurs de hachage.

Conclusion

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 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 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 cet article, 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

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

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

[3] : J. Kong, P. Zerfos, H. Luo, S. Lu, L. Zhang. «Providing robust and ubiquitous security support for mobile ad hoc networks». In Proceedings of the 9th International Conference on Network Protocols (ICNP), November 2001.

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

[5] : N. Asokan, P. Ginzboorg. «Key agreement in ad hoc networks". Computer Communications, 23:1627–1637, 2000.

[6] : Asad Amir Pirzada, Chris McDonald. «Kerberos Assisted Authentication in Mobile Ad-hoc networks». 27 th Australasian Computer Science conference in research and practice in Information Technology, Vol. 26. 2003.

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

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

[9]: Bruce Schneier, «Applied Cryptography», Second edition, 1996.

[10] : 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.

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

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

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