Sécurité dans les réseaux Ad-Hoc

Enseignants :
Yves GERARD
Laure GONNORD

Etudiants :
Mehdi HADDAD
Robleh WAIS

Université Lyon 1

Table des matières

I Introduction:

Considérons un réseau Ad-hoc. Un réseau Ad-Hoc est constitué de stations mobiles. Il faut donc pouvoir diffuser les données de manière sécurisée aux stations du réseau. Il faut prendre en compte que les stations se connectent et se déconnectent perpétuellement. Compte tenu de ces deux points, il faut pouvoir mettre en place un protocole permettant aux stations, connectées ou non, de récupérer les données échangées.

Dans le cryptage des données dans les réseaux ad-hoc, on peut distinguer deux catégories. Dans la première catégorie (« stateless »), on a une clé qui ne change jamais tout au long de la durée de vie du réseau. Et pour la deuxième catégorie (« stateful »), la clé peut changer à chaque fois qu'une station rejoint ou quitte le réseau. Dans ce second cas, il va falloir retransmettre la nouvelle clé aux stations en ligne. Contrairement à ce que l'on pourrait penser, c'est la deuxième catégorie qui est la moins coûteuse en coût de communication.

Les réseaux sans fils ont des caractéristiques particulières telles que la perte de connexion, faiblesse de la bande passante par moment et la mobilité des stations. Tous ces critères rendent le cryptage, des données, assez difficile à mettre en place. Il faut donc mettre en place des protocoles de diffusion de message aux stations qu'elle soit en ligne ou pas. On peut ainsi permettre aux stations n'étant pas en ligne au moment de la diffusion de récupérer le message diffusé lorsqu'elle n'était pas en ligne.

On propose un mécanisme permettant aux stations les plus éloignées de récupérer les messages échangés même en leur absence. On considère donc que chaque station conserve une partie du message. La station qui se reconnecte au réseau va pouvoir reconstituer le message en récupérant les bouts de message conservé par chaque station.

Il faut tenir compte que seules les stations en possession de ces clés peuvent décrypter les messages. Après exclusion d'une station, il faut procéder à un changement de clé et on la transmet aux utilisateurs restant dans le groupe.

Etant donné que les stations peuvent se déconnecter et se reconnecter, comment s'assurer qu'une station absente au moment de la mise à jour de la clé, reçoive la nouvelle clé?
Pour répondre à cette problématique, nous citerons les approches suivantes:

- Approche interactive : On demande au serveur de répéter les messages manqués (Peut demander beaucoup de ressources au serveur si les stations, ayant manqué les messages en question, sont nombreuses)
- Approche dite « naive » : Chaque station stocke tous les messages reçus. Quand une station se reconnecte, elle demande à ce qu'on lui envoie les messages, qu'elle a manqués. (Permet d'éviter la surcharge du serveur mais demande plus de mémoire au niveau des stations)
- Approche non interactive : Chaque station stocke une partie du message envoyé par le serveur au lieu de la totalité du message. Une station qui se reconnecte peut ainsi recomposer le message envoyé en récupérant les parties du message. (La surcharge du serveur est ainsi évitée et la mémoire demandée au niveau de chaque station est moindre)

II Présentation des réseaux Ad Hoc :

Un Réseau Mobile Ad Hoc ou "Mobil Ad Hoc Network" (MANET) est une collection d'unités mobiles munies d'interfaces de communication sans fil qui forment en coopérant, et d'une manière spontanée, un réseau temporaire. Ce réseau est réalisé indépendamment de toute infrastructure ou administration centralisée.

Les unités mobiles (nœuds, stations mobiles) sont équipées d'émetteurs et de récepteurs sans fil (exemple : les cartes WiFi qui peuvent atteindre des débits allant jusqu'à 54 MBits / seconde).

A un instant donné, selon la topologie du réseau (la position des nœuds, la configuration des antennes d'émission / réception, la puissance du signal et les interférences entre les canaux), une certaine connectivité existe entre les nœuds. Cette topologie change avec le temps en fonction des mouvements des nœuds ou tout autre changement de paramètre de configuration. La figure 1.1 montre comment la mobilité et la portée du champ de communication influent sur la nature des liens de transmission.

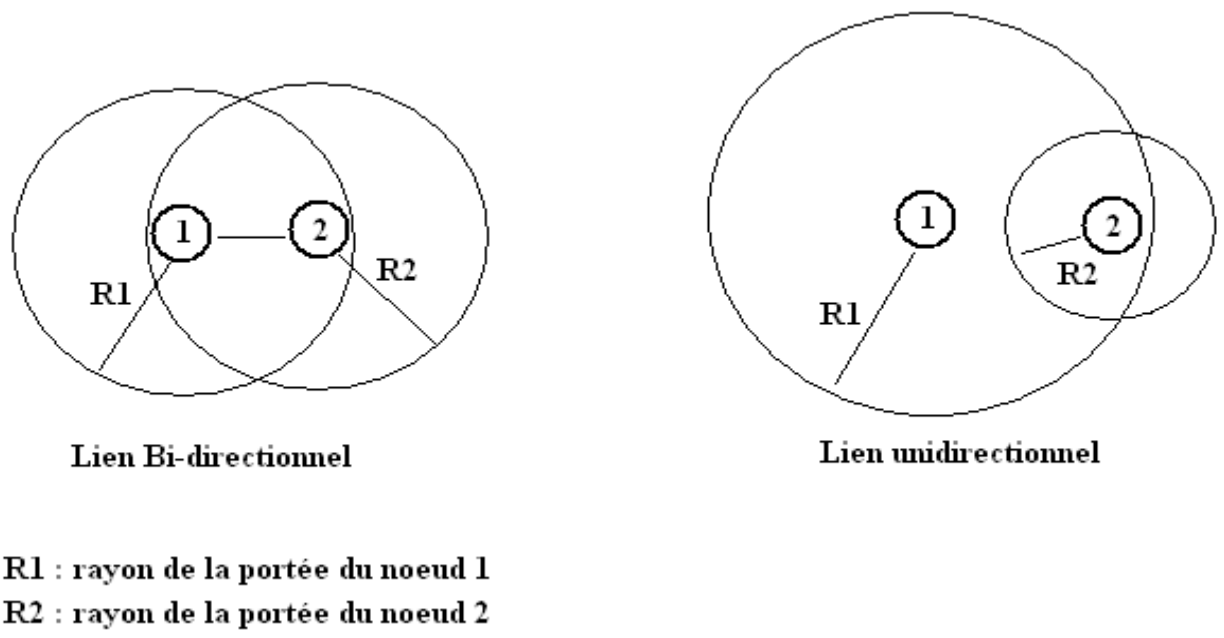

Figure 1.1 : Exemple de configuration d'un réseau ad-hoc

III Caractéristiques des réseaux Ad Hoc :

Description :

Un réseau ad-hoc est donc constitué d'entités, mobiles, qui communiquent entre elles. Chaque entité communique directement avec sa voisine. Pour communiquer avec d'autres entités, il lui est nécessaire de faire passer ses données par d'autres qui se chargeront de les acheminer.

Ainsi, le fonctionnement d'un réseau ad-hoc le différencie notablement d'un réseau comme le réseau GSM (Global System for Mobile communications).

Liens asymétriques :

En théorie, les liens sont symétriques car on a un affaiblissement du signal inversement proportionnel à la distance entre l'émetteur et le récepteur. Mais en pratiques ils sont asymétriques. On peut noter un déphasage dû aux multiples réflexions du signal sur différents obstacles. De plus, la route inverse n'est pas forcément la même que la route directe.

Sécurité limitée :

Vu les contraintes précédentes, les méthodes de sécurité (cryptage,... etc.) sont réduites ce qui augmente le risque d'attaques ou de piratage. En effet, les réseaux mobiles Ad-Hoc sont considérés comme étant très fragiles en matière d'attaques en tout genre. Lorsqu'une station émet des données, toute unité équipée d'un dispositif d'écoute (ici les cartes WiFi) a la possibilité d'intercepter ces données. Les pirates informatiques peuvent donc intercepter les données d'une manière directe en utilisant des antennes pirates (car les données circulent par voie hertzienne) ou bien obliger une station à consommer une bonne partie de ses ressources d'énergie en l'inondant de toutes sortes de requêtes inutiles.

Contraintes sur la bande passante :

Les réseaux sans fil se basant sur le partage des médiums de communication, alors la bande passante réservée à un hôte sera relativement modeste. Ceci implique des liens sans fil à capacité variable.

Pour les réseaux ad-hoc, les messages vont être cryptés. Nous avons également des clés « de groupe ». Nous savons que seules les stations en possession de clé peuvent décrypter les messages. Les mécanismes mis en place doivent permettre aux stations qui se sont déconnectée de pouvoir récupérer la ou les clés permettant les décryptages lors de leur re-connexion.

IV Protocoles

1. Les approches étudiées

- Approche dite « interactive »

En considérant que des stations déconnectées, au moment de l'envoi des messages, se reconnectent, elles ont donc manqué les messages envoyés lorsqu'elles étaient déconnectées. Elles vont donc demander au serveur de répéter les messages afin de pouvoir les récupérer.

Cette approche est simple est permet de répondre à la problématique. Par contre tout va sérieusement se compliquer si le nombre de stations, ayant manqué les messages, est important, le serveur va être énormément sollicité et les performances du réseau vont en être affectées.

L'approche interactive peut être intéressante pour un réseau de taille réduite. Elle peut être mise en place pour des réseaux domestiques.

- Approche dite « naïve »

Ici, les stations connectées vont stocker tous les messages qu'elles reçoivent. Elles vont donc avoir la possibilité d'envoyer les messages manqués aux stations « absentes ».

C'est une approche qui permet, elle aussi, de répondre à la problématique. De plus, la surcharge du serveur est évitée et les performances du réseau ne seront pas autant affectées. Mais, cette approche demande beaucoup plus de mémoire au niveau des stations. En effet, les stations devront stocker tous les messages envoyés. Lorsqu'il s'agit de quelques messages, il n'y a pas de problèmes. En revanche, dès que le nombre de messages est important, la mémoire sollicitée par les stations devient, elle aussi important et peut affecter les performances du réseau.

- Approche non interactive

Dans cette approche, chaque station va stocker une partie du message. Les stations « absentes » pourront donc récupérer le (ou les) message(s) manqué(s) en récupérant les « bouts » de message. De ce fait, les stations absentes pourront reconstituer le (ou les) message(s) manqué(s).

Cette approche résous le problème évoqué précédemment et améliore l'approche dites « naïve ». Le problème de la quantité de mémoire sollicité au niveau des stations est ainsi résolu. Il vaut mieux ne stocker qu'un morceau de message plutôt que de stocker le message tout entier.

C'est cette dernière approche qui va nous intéresser le plus, les autres étant des approches qu'on pourrait qualifier d'approche « préliminaires » qui ont permis d'arriver à cette dernière approche qui est la moins coûteuse en mémoire et qui affecte le moins les performances du réseau ad-hoc qui l'utilise.

2. *Approche non interactive*

Code Correcteur :

Cette approche se sert d'un code correcteur. Un code correcteur est une combinaison de deux algorithmes. Le premier est utilisé par l'émetteur pour coder son message, le second est utilisé par le récepteur pour décoder le message reçu et ainsi obtenir le message original.

Soit C=(Encode, Decode) un code correcteur. L'émetteur souhaite envoyer un message m composé de l symbole. Il commence par calculer c=Encode(m), et obtient donc un nouveau message composé de λ symboles qu'il transmet au récepteur.

L'avantage de l'utilisation d'un code correcteur réside dans le fait que le récepteur n'aura pas besoin de tous les symboles de c pour le reconstituer. En d'autre terme le récepteur aura besoin que d'un certain nombre d de symbole pour obtenir c. Une fois c reconstitué, il peut calculer Decode(c) et obtenir le message m.

Si le récepteur peut reconstituer un message à partir de n'importe quel (λ-d+1) symboles de c alors le code correcteur utilisé a une distance de longueur d.
Nous trouvons dans la littérature différent code correcteur. Les plus courants sont les code de Reed-Solomon [1], dans lequel un message peut être codé et décodé en $O(\lambda^2)$ et le code Tornado [2], dans lequel un message peut être codé ou décodé en $O(\lambda)$. Par contre le code Tornado ne garantit pas de pouvoir reconstituer le message de manière certaine mais donne une probabilité élevée de reconstituer le message.

Mécanisme de récupération de message :

Dans les réseaux ad hoc, la perte de message est assez fréquente. Elle est causée par les caractéristiques de ces réseaux tels que la déconnexion des nœuds, la partition temporaire du réseau, collision des messages...

Pour palier à ce problème il faut définir un mécanisme permettant, à un nœud, ayant manqué la transmission d'un message, de récupérer ce message.

Pour décrire ce mécanisme nous définissons les entités suivantes :

- N : l'ensemble des nœuds du réseau
- ON(t) : les nœuds en ligne (connectés) à l'instant t
- OFF(t) : les nœuds hors ligne (déconnectés) à l'instant t
- m(t) : message envoyer à l'instant t
- u : constante qui représente la durée qu'un nœud dans ON(t) stock une partie du message qu'il a reçu.

Donc un nœud $v \in OFF(t)$, qui se reconnecte à l'instant t', ne pourra récupérer que les messages m(t) avec $t' < t+u$.

Nous supposons, à présent, que la source S envoie un message m(t). Soit $v \in OFF(t)$ un nœud qui manque le message m(t) et supposons qu'à l'instant t' (t' > t) le nœud se reconnecte au réseau et souhaite récupérer le message m(t).

Soit C= (Encode, Decode) un code correcteur. Nous supposons qu'à chaque instant t il y a λ nœuds connectés (|ON(t)|>= λ).

Le mécanisme de récupération des messages manqué est le suivant :

- S diffuse c(t)= Encode (m(t))= (s(t,1), s(t,2), ..., s(t, λ)).
- Pour chaque nœud dans ON(t) :
  - Si (t>u) alors $\forall$ x $\in$ [1, λ] supprimer s(t-u, x)
  - Stocker s(t, y) avec y=Random(1, λ)
- Si à l'instant t' > t le nœud v $\in$OFF(t) veut récupérer m(t), il demande les symboles de c(t) à ses voisins jusqu'à en avoir assez de parties s(t,i) pour reconstituer c(t).
- v récupère m(t) = Decode(c(t)). Puis c(t)= Encode (m(t))= (s(t,1), s(t,2), ..., s(t, λ)). Et stock un symbole s(t, y) avec y = Random (1 , , λ)

La source souhaite envoyer un message m(t). Elle commence par calculer Encode(m(t)) et obtient un message c(t), ce message est composé de λ parties (s(t,1)... s(t, λ)). Le message c(t) possède la caractéristique, donnée grâce au code correcteur, qu'il peut être reconstitué à partir d'un sous ensemble de ses λ parties.

S envoie le message c(t) et chaque nœud, présent dans ON(t) qui reçoit le message, calcule Decode(c(t)) pour obtenir le message m(t). Chaque nœud ayant reçu le message stock une partie s(t,y) du message c(t). Il choisit la partie à stocker de manière aléatoire et comme tous les nœuds font leur choix de la même manière chaque partie sera stockée par un ou plusieurs nœuds. Après un certain temps u le nœud supprime la partie qu'il avait stocké, afin de ne pas surchargé sa mémoire.

Quand un nœud v $\in$OFF(t) se reconnecte et souhaite récupérer le message m(t) il demande à ses voisins les parties s(t,i) de c(t) jusqu'à en avoir assez pour reconstituer c(t). Une fois c(t) reconstitué il peut calculer Decode (m(t)) et aura donc récupérer le message raté. Puis v stock une partie de c(t) de manière aléatoire afin de pouvoir la transmettre à un nœud qui a, également, raté m(t) et se reconnectant après v.

Récupérer un seul message :

Nous souhaitons, à présent, déterminer le nombre de stations à interroger pour pouvoir récupérer un message manqué. Pour ce faire nous définissons l'ensemble Store(m(t),t') qui représente les nœuds qui ont stocké une partie du message m(t) (les nœuds présent à l'instant t et les nœuds qui ont manqué le message m(t) mais qui ont pu le récupérer et donc stocké, également, une partie du message).

Comme chaque nœud dans Store(m(t),t') choisit de stocker un partie du message de manière aléatoire, chaque partie du message se retrouve stockée par |ON(t)|/λ. Donc chaque partie sera dispersée de manière uniforme sur tout le réseau.

Avec l'utilisation d'un code correcteur, un nœud v ayant manqué un message m(t) n'aura pas besoin de toutes les λ parties du message. Mais λ-d+1 parties uniques lui suffisent pour reconstituer le message.

Ce problème de récupérer un ensemble de (λ-d+1) uniques parties à partir de λ peut se réduire au problème du coupon collecteur [3].

Soit X la variable aléatoire correspondante au problème. Et E(X) son espérance mathématique.

Nous avons alors :

$$E[X] \leq \lambda(\ln\lambda - \ln(d-1)) \leq \lambda \ln\left(\frac{\lambda}{d-1}\right) \leq \frac{l}{\rho} \ln\left(\frac{l}{\rho.(d-1)}\right)$$

Donc, en moyenne, un nœud ayant manqué un message m(t) aura besoin de O(l ln(l)).

Récupérer plusieurs messages :

Soit $(m(t_1), ...m(t_r))$ l'ensemble des messages manqués par un nœud $v \in OFF(t)$. Nous souhaitons savoir combien de stations, v devra interroger pour récupérer tous les messages.

En s'appuyant sur le résultat obtenu pour un seul message nous avons :

$$r.\frac{l}{\rho}\ln\left(\frac{l}{\rho.(d-1)}\right)$$

Notons qu'en pratique cette limite est plus petite car le nœud v peut récupérer plus partie de message différent d'un même nœud.

3. Utilisation du protocole

Communications point à point : le client et le serveur s'authentifient mutuellement en utilisant un protocole d'authentification ou un service d'authentification. A présent, ils peuvent construire en commun une clé symétrique et l'utiliser pour avoir une communication sécurisée.

Cette procédure peut être étendue à une communication d'une source à un groupe. La source aura avec chaque client une clé individuelle. Puis utilisera la clé de chaque client afin de lui retransmettre la clé du groupe. Ainsi, chaque client aura la clé du groupe et pourra donc communiquer avec le groupe de manière sécurisée

Supposons qu'un nœud souhaite faire partie du groupe il communique en mode point à point avec la source et aura donc sa clé individuelle. Puis, la source crée un nouvelle clé de groupe et la communique au nouveau membre en utilisant sa clé individuelle. Elle communique, également, cette nouvelle clé aux anciens membres en utilisant la clé de groupe précédente.

Le problème qui se pose est lorsque la source diffuse la nouvelle clé aux anciens membres il se peut qu'un membre manque cette mise à jour et donc sera exclu du groupe involontairement. Pour palier à ce problème nous pouvons utiliser le mécanisme de récupération de message décrit précédemment pour récupérer une ou plusieurs mises à jour manquées.

Récupération de mise à jour de clé manquée :

Une mise à jour de la clé de groupe peut être comme un message spécial. Nous utilisons donc le mécanisme de récupération de message décrit précédemment en utilisant un code correcteur avec les caractéristiques suivantes : un message m de longueur l sera codé en un message c de longueur 2l et le récepteur aura besoin de (l+1) symboles différents de c pour le reconstituer.

Si nous souhaitons connaitre le nombre de station à interroger pour qu'un nœud, ayant manqué une mise à jour, puisse la récupérer à sa reconnexion et donc pouvoir communiquer avec les membres du groupe.

En tenant compte des caractéristiques du code correcteur utilisé nous obtenons :

$$E[X] \leq \frac{2.k.\log(n).\ln(2)}{\sigma} \leq 1.3863.\frac{k.\log(n)}{\sigma}$$

Pour r clé manquées nous aurons :

$$E[X] \leq 1.3863.\frac{r.k.\log(n)}{\sigma}$$

Exemple :

Soit un réseau dont la topographie est décrite comme ceci: un serveur S1 et les entités (ou station) E1, E2, E3 et E4. Chaque entité a sa propre clé pour communiquer avec le serveur et le serveur à mis en place une clé de groupe permettant au à toutes les entités du groupe de pouvoir communiquer entre elles. Les informations échangées avec le serveur seront donc protégées. Dans le cas ou une entité est révoquée (suite à une utilisation malveillante par exemple), le protocole n'est pas très utile. Par contre lors qu'une nouvelle entité désire rejoindre le réseau, c'est la que le protocole étudié va être intéressant. En effet, considérons maintenant qu'une nouvelle entité E5 est acceptée dans le réseau alors que l'entité E3 n'était pas en ligne. S1 et E5 mettent en place leur clé individuelle, puis S1 va mettre en place une nouvelle clé de groupe. Cette clé de groupe va être diffusée à toutes les entités, présentes, de manière cryptée. Cette clé va être cryptée avec l'ancienne clé de groupe.

Après que la diffusion ait été réalisée, E3 est de retour sur le réseau. Or E3 , n'est pas en possession de la nouvelle clé de groupe. Comme nous considérons qu'une clé est un message particulier, les entités E1, E2 et E4 ont pu stocker un morceau de la nouvelle clé de groupe. Il ne reste plus à E3 d'interroger ses voisines afin de récupérer la nouvelle clé de groupe mise en place avec l'arrivée de l'entité E5.

La nouvelle clé de groupe étant cryptée avec l'ancienne clé de groupe, qui est en possession de E3, cette dernière va pouvoir la décrypter afin de pouvoir communiquer avec toutes les entités présentent dans le réseau.

V Conclusion

Les réseaux Ad-hoc sont de plus en plus répandus dans notre société. Le transfert de données doit pouvoir se faire en toute sécurité. Le mécanisme présenté lors de ce rapport permet un transfert de donnée sur et n'affectant pas trop les performances du réseau ou il est utilisé. C'est un mécanisme qui se montre plutôt robuste face aux attaques extérieures. Le fait de mettre en place, dans un réseau, un système de clé qui change à chaque inclusion ou révocation d'une station limite, aux « attaquants », les possibilités d'être en possession de la clé actuelle. Or quelqu'un qui tente d'infiltrer un réseau, ne peut décrypter les informations qui y sont échangée que s'il est en possession de la clé lui permettant d'effectuer ce décryptage.

Le mécanisme décrit permet de diminuer la surcharge globale du réseau. En effet, un nœud demandera, en priorité, à ses voisins les messages qu'il a manqué. De plus, la charge du serveur est moindre puisqu'il n'est pas sollicité par chaque ayant manqué un message.

Notons cependant que ce mécanisme en particulier les codes correcteurs sont vulnérable face aux « pollution attacks », qui fait partie des attaques par déni de service, ou un nœud mal veillant injecte des parties inexistante dans le message d'origine et ainsi rend la procédure de décodage plus difficile voir impossible.

Pour essayer de palier à ce genre d'attaques des « distillation code » peuvent être utilisés. Ces codes peuvent permettre de détecter les « vrai » partie du message envoyé.

Contribution

Nos connaissances concerneant le sujet du projet étant limitées, nous avons tous les deux travaillé ensemble sur toutes les parties du projet.

Références bibliographiques:

[1] Reed, I. et G. Solomon, Polynomial codes over certain finite fields, dans: Journal of the Society for Industrial and Applied Mathematics, 1960.

[2] Luby, M., M. Mitzenmacher, M. Shokrollahi et D. Spielman, Effcient erasure correcting codes, Transactions on Information Theory 47 (2001), pp. 569–584.

[3] Motwani, R. et P. Raghavan, “Randomized Algorithms,” Cambridge University Press, 1995 .

[4] Fiat, A. et M. Naor, Broadcast encryption, in: D. R. Stinson, editor, Advances in Cryptology – Crypto ’94, Lecture Notes in Computer Science **773** (1994), pp. 480–491.

[5] Wong, C., M. Gouda and S. Lam, Secure group communication using key graphs, in: ACM SIGCOMM ’98, 1998, pp. 68–79.