# Protection de la vie privée dans les réseaux mobiles ubiquitaires

Simon Boche

Université Rennes 1 - ISTIC

Supélec - Équipe CIDRE




# Protection de la vie privée dans les réseaux mobiles ubiquitaires

Boche Simon

Encadrants : Christophe Bidan, Sébastien Gambs, Nicolas Prigent

Master MRI

Rennes, le 5 juin 2012

# Table des matières

**Résumé**

La multiplication des services rend l'outil informatique de plus en plus omniprésent. De ce fait, l'informatique manipule de plus en plus de données nous concernant, données potentiellement sensibles. Ceci induit une augmentation importante des risques sur la vie privée des utilisateurs.

Parallèlement, les réseaux ad hoc, en interconnectant automatiquement et dynamiquement les machines des utilisateurs, permettent de fournir de nouveaux services sans nécessiter de déployer une infrastructure. Dans les réseaux ad hoc, chaque appareil joue le rôle de routeur et aide au transport des informations des autres nœuds. Ainsi, les informations sensibles des participants circulent via des canaux non contrôlés du réseau. De par leur mode de fonctionnement, les réseaux ad hoc diffusent une autre information très sensible du point de vue de la vie privée, la position des participants.

Afin d'améliorer le respect de la vie privée dans les réseaux ad hoc, nous proposons dans ce document le protocole NoName, prenant en compte le respect de la vie privée du point de vue de l'identité, des communications et de la position pour tous les nœuds du réseau.

Mots clés : Protection de la vie privée, réseaux ad hoc, informatique ubiquitaire, géolocalisation

# 1 Introduction

De plus en plus d'utilisateurs disposent d'un ordinateur portable, d'un téléphone intelligent (*smartphone*), d'une voiture connectée à Internet... Cette explosion d'appareils personnels mène à un nouveau modèle, l'informatique ubiquitaire. Ce terme désigne l'omniprésence de l'informatique dans le monde réel. Au contraire du modèle de l'ordinateur personnel, l'informatique ubiquitaire repose sur la communication et la coopération entre différents objets intelligents pour fournir des services augmentés.

L'informatique ubiquitaire, grâce notamment à la présence d'appareils mobiles, ouvre la voie à de nouveaux services, en particulier les services géolocalisés. Un service géolocalisé utilise la position d'un utilisateur (récupéré par une puce GPS par exemple) afin de fournir un service augmenté à l'utilisateur. Ce service augmenté peut être par exemple la découverte des restaurants à proximité de l'utilisateur ou la connaissance du trafic en temps réel permettant l'anticipation de ralentissements. De tels services existent à l'heure actuelle. Ainsi, l'entreprise Google propose une recherche géolocalisée, la position de l'utilisateur étant déterminée par son adresse IP. Afin d'améliorer l'expérience utilisateur et d'augmenter la pertinence des recherches, la future norme HTML 5 permet elle aussi l'intégration de la géolocalisation.

Les réseaux ad hoc sont une autre composante de l'informatique ubiquitaire. Ce type de réseaux diffère des réseaux classiques dans lesquels les équipements sont clairement identifiés. Dans un réseau ad hoc, il n'existe pas de point central où toutes les communications arrivent. Chaque objet est un routeur pour le réseau.

Le protocole de routage permet aux nœuds du réseau ad hoc de construire et de maintenir leur tables de routage. Du fait de la mobilité, les protocoles de routage ad hoc doivent être ainsi suffisamment réactif. En effet, un nœud peut disparaître du réseau ou encore changer de position et ne plus être relié aux mêmes voisins. Certains appareils mobiles disposant d'une capacité de calcul plus faible que les ordinateurs individuels, le routage ne doit donc pas être trop gourmand en ressources. De plus, la mobilité implique un fonctionnement sur batterie, l'autonomie des nœuds est donc primordiale.

L'informatique ubiquitaire est particulièrement sensible au risque de violation de la vie privée. La démocratisation de l'outil informatique et la compréhension très relative de l'outil par l'utilisateur ont permis la diffusion incontrôlée d'informations privées accessible par l'ensemble des internautes. Ainsi, début 2011, une partie de la base de données du PSN (PlayStation Network) contenant entre autre coordonnées bancaires et mails, s'est retrouvée accessible à tous.

La notion de vie privée est présente dans La Déclaration Universelle des Droits de l'Homme qui indique, dans son article 12, que "Nul ne sera l'objet d'immixtions arbitraires dans sa vie privée, sa famille, son domicile ou sa correspondance, ni d'atteintes à son honneur et à sa réputation. Toute personne a droit à la protection de la loi contre de telles immixtions ou de telles atteintes". En France, la loi informatique et liberté de 1978 encadre l'utilisation et la conservation d'informations personnelles.

Néanmoins, l'article 12 de la Déclaration Universelle des Droits de l'Homme reste assez flou sur la définition de la vie privée. Celle ci est en effet variable d'une personne à l'autre. Le débat actuel autour de l'utilisation de caméras vidéo sur la voie publique en est un bon exemple. De plus, suivant la situation, une même information peut être considérée comme relevant de la vie privée ou pas. Ainsi, la position géographique peut ne pas être considérée comme relevant de la vie privée. Toutefois, lorsqu'il est possible de relier cette information avec l'emplacement d'un cabinet médical par exemple, l'information rentre très clairement dans la sphère de la vie privée car elle permet déduire des informations médicales sur la personne. Par ailleurs, une information peut permettre d'inférer des données personnelles de manière indirecte (même si l'observateur en ignore la teneur). La simple existence d'une communication entre des personnes permet ainsi de dresser le graphe social d'une personne.

Dans la suite de ce document, nous commencerons par présenter les différents types de protocoles de routage ad hoc. Nous aborderons ensuite le respect de la vie privée et le lien très fort entre vie privée et géolocalisation. Nous continuerons avec une analyse de différents protocoles de routage ad hoc respectueux de la vie privée existants, puis nous détaillerons notre proposition de protocole et ses évolutions possibles.

# 2 Protocoles de routage ad hoc

Dans les réseaux fixes, le service de routage est assuré par des nœuds particuliers appelés routeurs qui font parti de l'infrastructure réseau. Dans un MANET, il n'existe pas d'infrastructure fixe. Le service de routage est assuré par l'ensemble des nœuds du réseau. Un protocole de routage ad hoc permet de créer les tables de routage. Il existe deux grands types de protocoles de routages ad hoc : les protocoles basés sur la topologie et ceux basés sur la position. Nous les présentons dans cette section.

## 2.1 Protocoles basés sur la topologie

Les protocoles basés sur la topologie représentent le réseau sous forme de graphe : deux nœuds sont dits voisins s'ils sont à portée radio l'un de l'autre. Un protocole de routage ad hoc se base sur cette notion de voisinage pour construire les tables de routage. La plupart des protocoles du monde filaire [1] sont des protocoles basés sur la topologie.

Il existe deux types de protocoles basés sur la topologie : les protocoles réactifs et les protocoles proactifs.

### 2.1.1 Protocoles proactifs

Un protocole proactif maintient à chaque instant dans sa table de routage un ou plusieurs chemins vers chaque autre nœud du réseau. Ainsi, chaque nœud peut à tout moment joindre un autre nœud du réseau. Pour maintenir ces informations à jour, des messages sont régulièrement envoyés pour mettre à jour les tables de routage. Cet échange de messages réduit la bande passante allouée aux données utiles mais permet d'avoir un temps de latence bas. Le protocole OLSR [6] est le protocole de routage ad hoc proactif le plus populaire.

Suivant OLSR, chaque nœud s'annonce à ses voisins en émettant un message de type HELLO et les détecte en écoutant les messages émis par ceux-ci. Les messages TC sont pour leur part diffusés dans le réseau et permettent de diffuser les informations de voisinage. En collectant les messages TC, un nœud peut ainsi construire de proche en proche le graphe du réseau est ainsi calculer la table de routage couvrant l'ensemble du réseau.

Pour limiter les messages TC envoyés, OLSR propose un système reposant sur des nœuds particuliers, appelés MPR (*MultiPoint Relay*). Seuls les MPR diffusent les messages TC reçus. Pour garder une couverture totale du réseau, un nœud choisi parmi ses voisins à un saut un sous-ensemble de nœuds (les nœuds MPR) capables de communiquer avec l'ensemble des nœuds à deux sauts.

OLSR est actuellement utilisé dans une réalisation d'un MANET sur téléphone Android, le projet Serval [2].

1. Comme le très connu BGP par exemple, utilisé pour le routage sur Internet
2. `http://www.servalproject.org/`

### 2.1.2 Protocoles réactifs

Les protocoles réactifs cherchent une route pour une destination que lorsque cela est nécessaire. Ainsi, lorsqu'un nœud A souhaite communiquer avec un nœud B, A regarde d'abord s'il connait une route pour atteindre B. Si ce n'est pas le cas, il envoie un message de découverte de route (*Route Request*, souvent abrégé RREQ) sur le réseau. Le message est diffusé dans le réseau jusqu'à a atteindre le nœud B. Celui-ci répond au message de découverte de route par un message de réponse (*Route Reply*, souvent abrégé RREP) permettant d'informer le nœud A de la route à suivre pour contacter le nœud B.

Un protocole réactif possède un avantage majeur : le réseau n'est sollicité que lorsqu'un message doit être diffusé et qu'aucune route n'existe. Ainsi, la bande passante utile est plus importante dans un protocole réactif que dans un protocole proactif. Malheureusement, la latence est aussi plus grande. En effet, la découverte de route peut prendre un certain temps : aucune donnée ne peut être envoyé avant de recevoir une réponse du nœud destinataire.

Le protocole DSR [10] est un exemple classique de protocole de routage ad hoc de type réactif. Dans DSR, le message de découverte de route contient la liste des nœuds traversés actuellement par le message. Lorsqu'un nœud autre que le nœud A et le nœud B reçoit ce message, il vérifie s'il a déjà traité le message. Cette vérification est nécessaire pour éviter la création de boucle de routage. S'il n'a pas déjà traité le message, il s'ajoute à la liste des nœuds dans le message et diffuse le message ainsi modifié. Lorsque le nœud B reçoit le message de découverte de route, celui-ci contient une route reliant le nœud A au nœud B. Il suffit donc d'utiliser cette route pour transmettre le message de réponse jusqu'au nœud A.

## 2.2 Protocoles de routage ad hoc basés sur la position

Le routage basé sur la position repose sur la localisation géographique des différents nœuds. Dans un protocole basé sur la position, chaque nœud est identifié par sa position. Il communique la sienne et s'informe sur celle des autres nœuds du réseau. Lorsqu'un nœud souhaite transmettre un message, il indique comme destination la zone géographique voulue. Cette information sert à transmettre le message vers le destinataire. Il existe différentes méthodes pour transmettre le message. L'approche MFR (*Most Forward within Radius*) consiste à comparer la position géographique de ses voisins à la zone géographique correspondant à la destination. Le message sera ensuite envoyé au voisin le plus proche de la destination. Une autre technique, appelée routage au compas (*Compass Routing*), utilise une ligne virtuelle reliant le nœud destinataire et le nœud source. Le prochain nœud à retransmettre le message sera le nœud voisin le plus proche de cette ligne. D'autres méthodes de routage géographiques sont expliquées dans [5].

Dans cette section, nous avons vu les deux grands types de protocoles de routage topologique : les protocoles de routage proactifs, permettant un temps de latence bas mais implique des échanges de messages très réguliers pour maintenir la table de routage à jour et les protocoles de routage réactifs, au temps de latence plus élevé mais moins consommateur de bande passante. Nous avons présenté un autre type de protocole, moins connu et moins utilisé : les protocoles basés sur la position. Ces protocoles utilisent leurs position comme identifiant et permet ainsi de communiquer entre des zones géographiques.

Dans un réseau de communication, la notion de position peut également relever de la vie privée. En effet, lorsqu'un sujet communique, il se trouve à une localisation donnée. S'il est possible de relier cette position géographique à un nœud, alors la vie privée n'est pas respecté.

# 3 Respect de la vie privée et géolocalisation

Le respect de la vie privée repose sur le lien existant entre une information sensible (le dossier médical par exemple) et l'identité de la personne concernée. C'est un domaine complexe : d'une personne à l'autre, une même information peut relever du domaine de la vie privée ou ne pas relever du domaine de la vie privée.

Le respect de la vie privée est un sujet très vaste. Nous nous concentrerons sur le respect de la vie privée dans les réseaux de communication. Dans cette section, nous présenterons d'abord un types d'attaques particuliers : les attaques sur les services géolocalisés. Nous verrons ensuite les propriétés relatives au respect de la vie privée. Enfin, nous présenterons un modèle permettant de représenter les différents types d'attaquants possibles.

## 3.1 Attaques sur les données géolocalisées

Il existe deux types de position, la position absolue et la position relative. La position absolue détermine les coordonnées précises d'un point sur la planète. Elle est le plus souvent décrite à l'aide de la longitude et de la latitude. Elle peut être récupérée en utilisant par exemple un récepteur GPS. La position relative, quant à elle, indique la proximité d'un point avec un autre point. Les trois paramètres permettant de décrire une position relative sont le point de référence, la direction et la distance. La position relative peut par exemple être déterminée en utilisant la topologie d'un réseau ad hoc. A partir de la position absolue d'un point A et de la position relative d'un point B faisant référence au point A, il est possible de calculer la position absolue du point B. En connaissant la position absolue d'un point A et d'un point B, il est possible de calculer la position relative du point B par rapport au point A.

Les attaques sur les données géolocalisées sont encore peu étudiés. Elle permettent cependant de récupérer des informations précieuses sur un utilisateur et ainsi porter atteinte à sa vie privée.

On entends par données géolocalisées des données constituées de l'identité de l'objet (une personne par exemple), de la position géographique et de l'heure. Ces informations permettent de reconstituer le parcours de l'objet. Au delà du parcours, ces informations permettent de déterminer des points d'intérêts. Un point d'intérêt est un lieu caractéristique de l'utilisateur, comme son domicile, son lieu de travail, ses magasins préférés... Ces points d'intérêts peuvent être extrait des données géolocalisées.

Une méthode simple consiste, à partir de données géolocalisées concernant une personne, à récupérer les deux points où la position géographique est resté la même durant une longue périodes de temps (typiquement quelques heures). Ces deux points sont généralement le domicile et le travail de cette personne. le *clustering*, une attaque par *inférence*, permet de grouper les données proches dans un même cluster et de séparer les données éloignées dans des clusters différents. Pour cela, il est nécessaire de pouvoir déterminer la proximité de deux objets (c'est à dire la distance relative les séparant). Une mesure simple de la distance entre deux objets dans le cas de données géolocalisées est la distance euclidienne entre les deux points mais des mesures plus complexes ont aussi été utilisées. Les attaques par inférences permettent aussi de deviner le prochain déplacement d'un sujet. Ces attaques reposent sur l'utilisation de chaine de Markov décrivant le comportement d'un sujet, créée à partir de données géolocalisées.

Un exemple d'implémentation d'attaques par inférences est le framework GEPETO [9]. Ce framework permet d'extraire de potentiels points d'intérêts à partir de données géolocalisées, suivant différents algorithmes. Cela permet ainsi de pouvoir comparer les différents algorithmes entre eux et de déterminer si le jeu de données géolocalisées porte atteinte à la vie privée des sujets.

Ainsi, la simple connaissance de la présence d'un nœud pendant un certain temps à un endroit précis est une information potentiellement sensible. Par exemple, si une personne participe au réseau ad hoc tout en étant dans un cabinet médical d'un spécialiste, il est raisonnable de penser que cette personne a des problèmes de santé. La position est donc une notion importante dans la problématique du respect de la vie privée.

### 3.2 Propriétés relatives au respect de la vie privée

Le respect de la vie privée est un sujet très vaste. Dans [15], Pfitzmann *et al.* définissent un certain nombre de notions liées au respect de la vie privée dans le contexte des réseaux de communication. Le modèle est formalisé de la façon suivante : des émetteurs envoient des messages à travers un réseau de communication à des receveurs. Les émetteurs et les receveurs sont des sujets du réseau de communication.

Dans ce contexte, un attaquant cherche à répondre la question suivante : qui (émetteur) communique avec qui (récepteur) ? L'action de communiquer est représenté, dans notre modèle, par un échange de message.

Une méthode pour protéger la vie privée consiste à assurer l'anonymat (c'est à dire ne pas pouvoir répondre à la question *qui*). La notion d'anonymisation n'est pas seulement lié à l'identité, mais à toute information permettant d'identifier de manière unique le sujet. Ainsi, selon Pfitzmann, l'anonymat d'un sujet signifie que ce sujet n'est pas identifiable dans un ensemble de sujets, appelé ensemble d'anonymat. On parle aussi de k-anonymité (le sujet se trouvant alors dans un ensemble d'anonymat composé de k éléments).

A partir de cette définition, il est possible de quantifier l'anonymat. Plus l'ensemble d'anonymat est grand, plus le sujet est anonyme. La taille de l'ensemble d'anonymat est maximale si elle concerne l'ensemble des sujets réalisant une même action (envoyer ou recevoir). De plus, l'ensemble d'anonymat n'est pas fixe dans le temps, mais change en fonction de l'activité des sujets.

Dans notre contexte, cette définition se décline pour les émetteurs et les receveurs. Ainsi, l'anonymat d'un émetteur signifie que cet émetteur n'est pas identifiable dans l'ensemble d'anonymat des émetteurs (et réciproquement pour les receveurs).De plus, dans un de réseau de communication ad hoc, un message est relayé par des nœuds intermédiaires. Le respect de la vie privée concerne donc également ces nœuds. Ainsi, nous définissons la notion d'anonymat pour un nœud intermédiaire de la façon suivante : un nœud intermédiaire est anonyme s'il n'est pas identifiable dans l'ensemble des nœuds intermédiaires.

La vie privée peut être relié à une autre définition : l'inassociabilité (*unlinkability*). L'inassociabilité de deux ou plus items d'intérêts (sujets, messages, actions...) d'un point de vue de l'attaquant signifie qu'il ne peut pas distinguer si ces items sont liés entre eux. Ainsi, nous pouvons parler d'inassociabilité entre un sujet et son identité ou entre un sujet et sa position géographique. Cette notion d'inassociabilité est centrale pour la compréhension du respect de la vie privée.

Dans notre contexte, nous parlons d'inassociabilité entre un sujet et un message. Ainsi, il n'est pas possible de relier un message à son émetteur ou le message à son destinataire. De plus, l'inassociabilité entre deux messages évite de pouvoir lier deux messages provenant d'un même émetteur et à destination du même receveur. En effet, s'il est possible de relier ces messages, nous pouvons avoir une idée du trafic entre les deux nœuds.

La section 3.1 nous indique l'importance de la position pour le respect de la vie privée. Nous pouvons ainsi décliner l'anonymat en deux parties : l'anonymat de l'identité et l'anonymat de la position. L'anonymat de l'identité concerne la protection de l'identité et réciproquement, l'anonymat de la position concerne la protection de la position.

Pour résumer, dans la suite de ce document, nous prendrons en considération les propriétés suivantes :

- Anonymat de l'identité de la source, de la destination et des nœuds intermédiaires
- Anonymat de la position géographique de la source, de la destination et des nœuds intermédiaires

– Innassociabilité entre l'émetteur d'un message et le message
– Innassociabilité entre le destinataire d'un message et le message
– Innassociabilité entre deux messages émis par le même émetteur et à destination du même destinataire

## 3.3 Modèles d'attaquants

La sécurité d'une solution est analysée par rapport à un type d'attaquant donné. On parle d'attaquant plus fort qu'un autre lorsque ses capacités représentent une menace plus importante pour la sécurité ou le respect de la vie privée des entités du système. Il existe différents critères discriminant les attaquants sur un réseau ad hoc respectueux de la vie privée.

### L'activité de l'attaquant

Un attaquant dispose de plusieurs méthodes pour attaquer un réseau. Il peut par exemple écouter les messages échangés, les intercepter ou bien envoyer de faux messages sur le réseau. Cela nous permet de distinguer deux types d'attaquant : les actifs et les passifs.

Un attaquant passif ne fait qu'écouter les messages, au contraire de l'attaquant actif qui, lui, peut envoyer des messages (qui potentiellement peuvent perturber le bon fonctionnement du protocole de routage).

### La zone de couverture de l'attaquant

Un attaquant peut avoir une vision plus ou moins grande du réseau. Ainsi, nous distinguons deux types d'attaquants : les attaquants globaux et les attaquants locaux. Un attaquant global est un attaquant ayant une vision totale de ce qui se passe sur le réseau ad hoc. Un attaquant local, au contraire, va avoir une vision partielle du réseau (de la taille de sa zone de transmission et de réception).

L'attaquant collusif est situé entre l'attaquant local et l'attaquant global. Cet attaquant a à sa disposition plusieurs appareils collaborant entre eux et lui offre une vision partielle du réseau.

Notons aussi que la position de l'attaquant a elle même de l'importance. Ainsi, un attaquant se trouvant sur une route entre deux nœuds ciblés aura accès à plus d'informations et sera donc plus fort qu'un attaquant en dehors de la route.

La combinaison de ces différents critères permet de définir un attaquant. Par exemple, nous pouvons considérer un attaquant global et passif : cet attaquant a une vision globale du réseau et ne fait qu'intercepter les messages émis, sans en émettre lui même.

Dans la suite de ce document nous nous intéressons à deux types d'attaquants : **l'attaquant global passif** et **l'attaquant collusif actif**.

# 4 Étude de protocoles de routage ad hoc respectueux de la vie privée

Le routage respectueux de la vie privée dans les réseaux ad hoc est un sujet déjà abordé dans la littérature. Nous analysons dans cette partie sept protocoles de routage. Chaque analyse se découpe de la façon suivante : l'introduction explique les concepts généraux utilisés dans le protocole. Une description rapide du fonctionnement du protocole est ensuite fournie. Les vulnérabilités et les attaques possibles sont ensuite détaillées. Enfin, nous terminons par évaluer les performances du protocole d'un point de vue réseau.

A l'exception d'ALARM, tous les protocoles étudiés sont des protocoles réactifs. De manière générale, tous les protocoles font les hypothèses suivantes :

- Les liens entre les nœuds sont symétriques. Si un nœud A peut envoyer un message avec le nœud B, alors le nœud B peut envoyer un message au nœud A. L'ensemble des nœuds du réseau forme ainsi un graphe non-orienté.
- Tous les protocoles étudiés utilisent de la cryptographie. Les nœuds doivent donc disposer des ressources nécessaires pour utiliser des fonctions cryptographiques.
- Les ressources d'un attaquant ne sont pas illimités (en temps et en mémoire). Notamment, il ne peut casser les algorithmes de chiffrement en testant toutes les clés possibles.

Nous étudierons tous d'abord différents protocoles basés sur la topologie puis les protocoles basés sur la position.

## 4.1 Protocoles basés sur la topologie

Nous analyserons dans cette section 5 protocoles de routage topologique : ANODR, ASR, SDAR, ARM et enfin MASK.

### 4.1.1 ANODR

ANODR [11] utilise le concept de la trappe (*trapdoor*) et du routage en oignon pour assurer les propriétés de respect de la vie privée.

Une trappe est un système permettant de cacher l'identité réelle du destinataire. Plus formellement, c'est une fonction à sens unique, que seul le destinataire, ayant une information supplémentaire, peut inverser. "L'ouverture" d'une trappe consiste à tenter d'inverser la fonction. Une implémentation possible d'une trappe consiste à utiliser le chiffrement asymétrique. Ici, chaque nœud $i$ possède une clé secrète $KS_i$ et une clé publique $KP_i$.

On appelle routage en oignon un routage utilisant un oignon cryptographique. Un oignon cryptographique est un message chiffré successivement par un ensemble de clés (ici $K_a$, $K_b$ et $K_c$)

$$\{\{\{msg\}_{K_a}\}_{K_b}\}_{K_c}$$

Chaque clé est propre à un routeur. Ainsi, seul le routeur possédant la clé $K_a$ peut déchiffrer la première couche de l'oignon. En empilant les couches cryptographiques, l'oignon cryptographique contient la route que va traverser le message. Dans ANODR, l'oignon cryptographique va être construit au fur et à mesure de la propagation d'un message de découverte de route. Il va ensuite être utilisé par la destination pour contacter la source.

L'ensemble des nœuds sur la route est numéroté de 0 à $n$, avec 0 le nœud source et $n$ le nœud destinataire. Les auteurs supposent que chaque nœud $i$ du réseau ad hoc possède une clé symétrique que lui seul connait, notée $K_i$.

**Déroulement du protocole**

Lorsque le nœud source souhaite découvrir un chemin pour une destination donnée, il envoie un paquet de la forme

$$< RREQ, Seqnum, tr_{dest}, onion_0 = \{N_0\}_{K_0} >$$

$Seqnum$ correspond à un numéro de séquence global unique. $tr_{dest}$ est la trappe que seul le nœud destinataire peut ouvrir. $onion_0$ correspond à l'oignon envoyé par le nœud 0. $N_0$ est un nombre aléatoire généré par le nœud 0.

Le champ $tr_{dest}$ est construit de la façon suivante

$$tr_{dest} = \{n, N_{tr}\}_{KP_n}$$

Avec $N_{tr}$ un nombre aléatoire et $n$ l'identité de la destination.

Le nœud intermédiaire $i$ reçoit le message de type $RREQ$ suivant

$$< RREQ, Seqnum, tr_{dest}, onion_{i-1} = \{N_{i-1}, \{N_{...}, \{N_0\}_{K_0}\}_{K_{...}}\}_{K_{i-1}} >$$

Il teste tout d'abord s'il a déjà traité le message. Pour cela, il compare le numéro de séquence $Seqnum$ avec les numéros de séquences des messages qu'il a déjà vu passer. S'il a déjà traité le message, il le supprime. Sinon, il enregistre le numéro de séquence dans un cache.

Il peut ensuite tester la trappe. S'il n'est pas le destinataire, il retransmet le message $RREQ$ reçu en modifiant le champ onion

$$< RREQ, Seqnum, tr_{dest}, onion_i = \{N_i, onion_{i-1}\}_{K_i} >$$

Puis il enregistre $N_i$ dans une table.

Lorsque le nœud destinataire $n$ reçoit le message $RREQ$, il transmet un message $RREP$ de la forme

$$< RREP, P_n, pr_{dest} = N_{tr}, onion_{n-1} >$$

Avec $P_n$ le pseudonyme global unique du nœud $n$, $pr_{dest}$ la preuve de l'ouverture de la trappe, et le champ $onion_{n-1}$ du paquet $RREQ$ émis par le nœud $n-1$. Ce champ contient la route utilisée par le message $RREQ$ pour arriver à destination.

Table de routage : $\boxed{Pi}\boxed{Pi+1}$

Cache : $\boxed{Seqnum}$

FIGURE 1 – Tables de routage pour le protocole ANODR

Le nœud intermédiaire $i$ reçoit un message $RREP$ de la forme

$$< RREP, P_{i+1}, pr_{dest}, onion_i >$$

Il déchiffre dans un premier temps le champ $onion_i$ avec sa clé personnelle $K_i$. Il compare ensuite le résultat obtenu avec le nombre aléatoire $N_i$, stocké dans une table. Si le résultat est correct, cela signifie que le nœud $i$ se trouve bien sur la route entre le nœud $n$ et le nœud 0. Il remplace alors le pseudonyme $P_{i+1}$ par son pseudonyme $P_i$ et enregistre dans sa table de routage les informations nécessaires (voir figure 1). De plus, il remplace le champ $onion_i$ par le champ $onion_{i-1}$, obtenu lors du déchiffrement d'$onion_i$. Il transmet ensuite le message ainsi modifié

$$< RREP, P_i, pr_{dest}, onion_{i-1} >$$

Le nœud source 0 reçoit le message $RREP$ suivant

$$< RREP, P_1, pr_{dest}, onion_0 >$$

Tout comme un nœud intermédiaire, le nœud source déchiffre $onion_0$ et vérifie si le résultat fourni est égal à $N_0$. Si c'est le cas, il est bien le destinataire du message $RREP$. Il vérifie ensuite l'ouverture de la trappe à l'aide de $pr_{dest}$. Cette vérification permet au nœud source de s'assurer que la route créée est bien à destination du nœud $n$. En effet, seul le nœud destination peut déchiffrer la trappe et apporté la preuve de l'ouverture au nœud source.

Lorsque le nœud source souhaite envoyer un message au nœud destinataire, il lui suffit de mettre en entête du message le pseudonyme $P_1$ puis d'envoyer le message. A l'aide de sa table de routage, le nœud 1 peut déterminer que le message est pour lui et à quel nœud il doit transmettre le message. Il remplace alors le pseudonyme $P_1$ par le pseudonyme $P_2$. Les nœuds intermédiaires procèdent aux mêmes opérations, permettant ainsi d'envoyer de proche en proche un message de la source vers la destination. Le principe est similaire pour envoyer un message du destinataire à la source.

**Propriétés de respect de la vie privée**

Dans ANODR, l'identité de l'ensemble des nœuds du réseau est assuré. En effet, l'identité des nœuds présents sur la route est remplacée par un nombre aléatoire. L'identité de la destination est protégée en utilisant une trappe.

Le respect de la vie privée d'un point de vue de la position n'est par contre pas assuré par ANODR. En effet, un attaquant collusif actif peut utiliser une attaque par rejeu. Il suffit pour cela d'intercepter une trappe et de la rejouer. En écoutant le trafic, il est possible de déterminer dans quelles zones ne se trouvent pas le destinataire. En rejouant suffisamment de fois, il est possible de réduire fortement l'anonymat de nœud destinataire.

Un attaquant collusif actif peut aussi utiliser le champ *onion* pour déterminer la position relative d'un nœud et plus particulièrement du nœud source et du nœud destinataire. Il suffit pour cela de regarder la taille du champ. Celui-ci grossit au fur et à mesure de la traversée du message $RREQ$ et diminue au fur et à mesure de la traversée du message $RREP$. Cette taille permet ainsi de déterminer le nombre de sauts entre un nœud attaquant et le nœud source ou entre un nœud attaquant et le nœud destinataire.

Un attaquant global passif peut réaliser la même attaque sur la position des nœuds qu'un attaquant collusif local. En effet, la taille du champ *onion* est disponible pour l'ensemble des nœuds du réseau. Un tel attaquant peut aussi suivre à la trace les messages de type $RREP$. En effet, le champ $pr_{dest}$ est fixe. Cela permet à l'attaquant de trouver les nœuds intermédiaires ainsi que le nœud source et le nœud destination.

Une autre méthode pour tracer les messages de type $RREP$ repose sur l'observation du champs *onion*. Celui-ci est construit au fur et à mesure pendant la transmission du message $RREQ$. En enregistrant ces messages et en comparant les champs *onion* entre les messages $RREQ$ et $RREP$, un attaquant global peut retrouver l'ensemble des nœuds participant à une route.

Une autre technique pour un attaquant global externe consiste à déterminer dans quelle zone géographique s'arrête un message de type $RREQ$ (zone où se trouve donc le destinataire) et dans quelle zone s'arrête la transmission d'un message $RREP$ (zone où se trouve l'expéditeur).

L'inassociabilité n'est pas assuré par ANODR. En effet, Sans utiliser de données contenues dans les messages, il reste possible de tracer un message. Cette attaque repose sur l'utilisation du temps comme canal caché. Cela consiste à relier un message arrivant sur un nœud avec un message sortant de ce nœud quelques millisecondes plus tard. Il est probable que l'arrivée du premier message soit la cause de l'envoi du second. Ainsi, il est possible en remontant la route de retrouver l'émetteur d'un message ou le destinataire du message. De plus, l'inassociabilité entre deux messages n'est pas assuré. En effet, par construction, tous les messages diffusés sur une route (donc avec les pseudonymes temporaires choisis pour celle-ci) sont à destination d'un nœud et provenant de la même source.

**Performances**

Les performances d'ANODR n'ont pas été suffisamment analysées par les auteurs. Néanmoins, nous savons qu'ANODR utilise de façon intensive le chiffrement et le déchiffrement, et cela pour tous les nœuds du réseau lors d'un message $RREQ$ et pour tous les nœuds participant à la route pour un message $RREP$. Cela implique un temps de latence incompressible sur chaque nœud du réseau. De plus, les messages ne sont pas de taille fixe. Ainsi, le champ *onion* grandit au fur et à mesure de la traversée du réseau. Cela nécessite donc un temps de transmission plus important et donc une consommation d'énergie elle aussi plus importante.

Notons qu'un autre protocole, le protocole AnonDSR [19], très proche d'ANODR, tente de corriger le problème de temps de latence et plus largement de performances en utilisant un système d'index permettant d'éviter de tester l'ensemble des clés possédés par un nœud.

ANODR, par le système de trappe, permet de protéger l'identité du nœud destinataire. Malheureusement, il ne prend pas en compte l'anonymat de la position pour la source et la destination. De plus, l'utilisation du routage en onion provoque des performances dégradées.

### 4.1.2 ASR

Comme ANODR, Le protocole ASR [23] repose sur l'utilisation d'une trappe permettant de protéger l'identité du destinataire. L'originalité du protocole est de proposer une structure de données permettant de remplacer un champ TTL (*Time To Live*) tout en empêchant les nœuds intermédiaires de déterminer la position relative du nœud source. Traditionnellement, un TTL permet de détecter des boucles de routage et de contrôler la distance parcourue par le message. Le TTL est aisé à mettre en œuvre : un compteur est initialisé par l'expéditeur du message. Chaque nœud traitant le message décrémente de un le compteur. Si le compteur arrive à zéro, le message est détruit. Malheureusement, le TTL permet de connaitre le nombre de sauts (et donc la position relative) entre le nœud recevant le message et le nœud source. Les auteurs proposent d'utiliser un TTL sécurisé : seul le nœud destinataire peut connaitre le nombre de sauts entre lui et la source. Chaque nœud intermédiaire va décaler de façon circulaire le champ TTL de $p$ bits (les $p$ derniers bits devenant les $p$ premiers bits). En connaissant le champ original, il est possible de déterminer le nombre de sauts entre la source et la destination.

Une autre particularité du protocole est d'utiliser une fonction à sens unique "rapide". Cette fonction (notée $H(key, data)$, $key$ étant une clé et $data$ les données) permet de tester rapidement si un nœud doit router un message. Après la phase de découverte de route, les nœuds voisins ont établi deux à deux une clé partagée. En utilisant un challenge (c'est à dire, vérifier la connaissance de la clé par les deux nœuds), le nœud peut déterminer qu'il partage une clé avec le nœud lui envoyant le message. Il doit donc transférer le message.

Les auteurs supposent que les nœuds partagent deux à deux une clé partagée. $K_{i,j}$ est la clé partagée par le nœud $i$ et le nœud $j$.

**Déroulement du protocole**

L'ensemble des nœuds sur la route est numéroté de 0 à $n$, 0 étant le nœud source et $n$ le nœud destinataire. $U_i$ représente la structure de donnée générée par le nœud $i$ et $KP_i$ une clé publique générée par le nœud $i$.

Le TTL sécurisé est construit de la façon suivante. $U_0$ est un nombre aléatoire généré par le nœud source avec $size(U_0) = (nombre\_de\_saut\_maximum + 1) * p$. $U_i$ est généré de la façon suivante

$$U_i = (U_{i-1} \oplus N_i) >> p$$

$N_i$ est un nombre aléatoire choisi par le nœud $i$ et de taille $p$. L'opération $\oplus$ représente un XOR (ou exclusif). L'opération XOR va donc modifier les $p$ derniers bits de $U_{i-1}$. Puis le nœud effectue un décalage à droite circulaire de $p$ bits (les $p$ derniers bits devenant les $p$ premiers bits).

Le nœud S envoie un paquet $RREQ$ de la forme :

$$< RREQ, seq, \{n, K_s, U_0\}_{K_{0,n}}, \{seq\}_{K_s}, KP_0, U_0 >$$

$seq$ correspond au numéro de séquence. $K_s$ est une clé de session choisi par le nœud source. Celle-ci est chiffré à l'aide de la clé partagée $K_{0,n}$. $KP_0$ est une clé publique temporaire choisis par la source.

Un nœud intermédiaire $i$ reçoit le message $RREQ$ organisé de la manière suivante :

$$< RREQ, seq, \{n, K_s, U_0\}_{K_{0,n}}, \{seq\}_{K_s}, KP_{i-1}, U_{i-1} >$$

Le nœud $i$ vérifie dans un premier temps s'il a déjà traité le message, en comparant celui-ci à ceux stockés dans un cache. S'il a déjà traité le message, il le supprime. Sinon, le nœud vérifie s'il est le destinataire du message en ouvrant la trappe (ici, $\{n, K_s, U_0\}_{K_{0,n}}$). Pour l'ouverture de la trappe, le nœud intermédiaire va tester l'ensemble des clés partagées qu'il possède avec les autres nœuds du réseau. S'il n'est pas le destinataire du message, il enregistre $seq, KP_{i-1}, \{seq\}_{K_s}$ dans le cache. Il génère ensuite une clé publique temporaire $KP_i$ et génère $U_i$, puis il envoie le message suivant

$$< RREQ, seq, \{n, K_s, U_0\}_{K_{0,n}}, \{seq\}_{K_s}, KP_i, U_i >$$

Le destinataire reçoit le message suivant

$$< RREQ, seq, \{n, K_s, U_0\}_{K_{0,n}}, \{seq\}_{K_s}, KP_{n-1}, U_{n-1} >$$

Il vérifie tout d'abord s'il a déjà traité le message puis s'il est le destinataire. Une fois ces vérifications faites, il compare $U_{n-1}$ et $U_0$ en décalant à droite de $p$ bits $U_{n-1}$ et $U_0$ autant de fois nécessaire pour avoir $U_{n-1} = U_0 \neq 0$. Le nombre de décalage correspond au nombre de nœuds situés entre le nœud source et le nœud destination. Si la comparaison échoue, alors le nombre de sauts à dépassé la valeur de $nombre\_de\_sauts\_maximum$ et le message est supprimé. Sinon, le nœud destinataire $n$ envoie le message suivant

$$< RREP, \{N_n\}_{KP_{n-1}}, \{seq, K_s\}_{N_n} >$$

$N_n$ est un nombre aléatoire généré par le nœud destinataire $n$. $KP_{n-1}$ correspond à la clé publique générée par le nœud $n-1$ et présent dans le message $RREQ$ reçu par le destinataire.

Le nœud intermédiaire $i$ reçoit le message de type $RREP$ suivant

$$< RREP, \{N_{i+1}\}_{KP_i}, \{seq, K_s\}_{N_{i+1}} >$$

Le nœud $i$ déchiffre le premier champ du message en utilisant sa clé secrète $KS_i$. Si le déchiffrement échoue, le nœud supprime le message. Sinon, il utilise $N_{i+1}$ pour déchiffrer le deuxième champ. Il vérifie ensuite si le message $RREP$ vient bien du destinataire du message. Pour cela, le nœud chiffre $seq$ avec la clé de session $K_s$. Il compare ensuite ce résultat au champ $\{seq\}_{K_s}$ du message $RREQ$ ayant pour numéro de séquence $seq$ (ces informations sont disponibles dans le cache du nœud). Si la comparaison réussit, le nœud $i$ génère un nombre aléatoire $N_i$ et envoie le message suivant

$$< RREP, \{N_i\}_{KP_{i-1}}, \{seq, K_s\}_{N_i} >$$

et enregistre dans sa table de routage (voir figure 2) les informations suivantes : $seq$, $KP_{i-1}$, $N_i$, $N_{i+1}$.

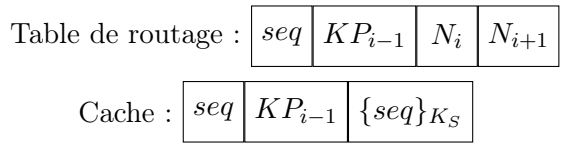


FIGURE 2 – Table de routage du protocole ASR

Le nœud source reçoit le message $RREP$ suivant

$$< RREP, \{N_1\}_{KP_0}, \{seq, K_s\}_{N_1} >$$

et réalise le même traitement qu'un nœud intermédiaire, mise à part la transmission d'un nouveau message.

Lorsque le nœud $i$ souhaite transmettre un message au nœud $i+1$, il préfixe le message par le champ $TAG$, composé de la façon suivante

$$TAG_i =< Nounce_i, H(N_{i+1}, Nounce_i) >$$

$Nounce_i$ est un nombre aléatoire choisi par le nœud $i$. Lorsque le nœud $i+1$ reçoit le message, il vérifie le champ $TAG$ en utilisant la fonction $H$. Si la vérification est incorrecte, le nœud supprime le message. Sinon, le nœud regarde dans sa table de routage, génère $TAG_{i+1}$ et diffuse le message. Les données peuvent aussi être chiffrées à l'aide du secret partagé.

**Propriétés de respect de la vie privée**

ASR assure la protection de l'identité de l'ensemble des nœuds du réseau en utilisant des clés publiques générées à la volée. Ainsi, à aucun moment l'identité du nœud n'est diffusée sur le réseau.

Le protocole n'assure par contre pas la protection de la position des nœuds. Des attaquants collusifs actifs peuvent découvrir la position relative du nœud source. Soit $x$ et $y$ deux nœuds du réseau situé à une nombre de sauts connu des deux nœuds. En comparant $U_x$ et $U_y$, ils déterminent qu'elle est la partie commune. En connaissant la taille des champs $U_{0,\ldots,n}$, ils peuvent ainsi connaitre le décalage nécessaire pour que $size(U_x \cap U_y) = size(U_0)$ et ainsi connaitre le nombre de sauts pour atteindre le nœud source.

De plus, un attaquant global passif peut utiliser la technique de comparaison des TTL sécurisés en interceptant l'ensemble des champs $U_{0,\ldots,n}$ et, ainsi, recréer la topologie du réseau ad hoc. Ainsi, la position du nœud destinataire et les positions nœuds intermédiaires ne sont pas protégées.

ASR est aussi sensible à une attaque basé sur le temps. En effet, un message arrivant sur un nœud est la conséquence du message sortant du nœud. Il est ainsi possible pour un attaquant global de retrouver le nœud source et le nœud destinataire. Cela implique que l'inassociabilité de la source et d'un message et l'inassociabilité de la destination et d'un message ne sont pas assurées.

L'inassociabilité d'un message par rapport à un autre message n'est pas non plus assurée. En effet, tout comme pour ANODR, tout messages passant par cette route implique une communication entre le nœud source et le nœud destinataire.

**Performances**

ASR utilise le chiffrement asymétrique pour la trappe et pour les messages de type $RREP$. Cela sous-entend une tentative de déchiffrement par l'ensemble des nœuds. Or, le déchiffrement asymétrique est lent et provoque une latence lors de la découverte de route.

L'utilisation du champ $TAG$ utilisant la fonction à sens unique rapide permet néanmoins d'accélérer la transmission des données par rapport à la phase de découverte de route en diminuant le temps nécessaire au nœud pour déterminer si un message doit être routé ou pas.

En conclusion, le TTL sécurisé ne permet pas de protéger la position de la source de manière correcte face à un attaquant collusif. De plus, un attaquant global peut recréer l'ensemble de la topologie du réseau, simplement en écoutant le trafic. ASR, bien que protégeant l'identité de la source et de la destination, ne permet donc pas de respecter l'anonymat de la position des nœuds participant au réseau.

### 4.1.3 SDAR

Comme ANODR, SDAR [3] utilise un routage en oignon et une trappe. De plus, le protocole propose un système de réputation.

Un système de réputation consiste à noter les différents participants sur leur comportement. Dans le protocole SDAR, le système de notation évalue le respect du protocole par les nœuds du réseau. Trois niveaux de confiance sont créés : confiance faible, confiance moyenne et confiance forte. Un nœud peut décider, lors de l'envoi de la découverte de route, du niveau de confiance voulue pour la communication.

L'originalité du protocole est sur l'utilisation du routage en oignon. En effet, l'oignon cryptographique n'est pas construit au fur et à mesure mais par le destinataire. Cela nécessite que les nœuds intermédiaires envoient les informations nécessaires pour la construction de cet oignon cryptographique.

Le protocole fonctionne sur le principe de cellule (c'est à dire les nœuds voisins). Les nœuds annoncent leurs présences en envoyant des messages HELLO. Un message envoyé par $i$ contient la clé publique $KP_i$ associée à la clé privée $KS_i$ du nœud $i$ faisant parti d'une cellule. Dans celle-ci, un nœud central génère les clés correspondant aux différents niveaux de confiance. Il s'occupe ensuite de la diffusion de ces clés suivant le niveau de confiance de chaque nœud.

Les auteurs supposent l'existence d'une autorité de certification (CA) à l'extérieur du réseau ad hoc. De plus, chaque nœud du réseau peut être identifié par son adresse IP sur le réseau ad hoc.

Dans la suite, nous considérons que les échanges se réalisent entre des nœuds de même confiance. Nous ne faisons donc pas apparaitre le chiffrement de la cellule dans les messages.

**Déroulement du protocole**

Le protocole de découverte de route est le suivant : le nœud source 0 envoie un message de type $RREQ$

$$< RREQ, trust, KP_{temp}, \{n, K_s, size(padding)\}_{KP_n}, padding, \\ \{0, KP_0, KP_{temp}, KS_{temp}, N_0, \{msg_0\}_{KS_0}\}_{K_s} >$$

$trust$ correspond au niveau de confiance que veut le nœud source pour cette communication. Ce champ peut prendre les valeurs suivantes : $LOW$, $MEDIUM$ et $HIGH$. $KP_{temp}$ et $KS_{temp}$ sont respectivement une clé publique et une clé privée temporaire.$K_s$ est une clé symétrique de session. $msg_0$ correspond au message $RREQ$ émit par le nœud 0.

Le nœud intermédiaire $i$ reçoit le message $RREQ$ suivant

$$< RREQ, trust, KP_{temp}, \{n, K_s, size(padding)\}_{KP_n}, padding, \\ \{0, KP_0, KP_{temp}, KS_{temp}, N_0, \{msg\}_{KS_0}\}_{K_s}, \\ \{1, K_1, N_1, \{msg_1\}_{KS_1}\}_{KP_{temp}}, \\ \dots \\ \{i-1, K_{i-1}, N_{i-1}, \{msg_{i-1}\}_{KS_{i-1}}\}_{KP_{temp}} >$$

Le nœud intermédiaire vérifie dans un premier temps s'il a déjà traité le message. Pour cela, il utilise $KP_{temp}$ comme numéro de séquence unique. Si le nœud a déjà traité le message, il le supprime. Sinon, $i$ vérifie s'il est le destinataire du message en tentant de déchiffrer le troisième champ du message $RREQ$. S'il n'est pas le destinataire, il génère une clé $K_i$ et un nombre aléatoire $N_i$. Il diffuse ensuite le message suivant (chiffré avec la clé commune adéquate)

$$< RREQ, trust, KP_{temp}, \{n, K_s, size(padding)\}_{KP_n}, padding, \\ \{0, KP_0, KP_{temp}, KS_{temp}, N_0, \{msg\}_{KS_0}\}_{K_s}, \\ \{1, K_1, N_1, \{msg_1\}_{KS_1}\}_{KP_{temp}}, \\ \dots \\ \{i-1, K_{i-1}, N_{i-1}, \{msg_{i-1}\}_{KS_{i-1}}\}_{KP_{temp}}, \\ \{i, K_i, N_i, \{msg_i\}_{KS_i}\}_{KP_{temp}} >$$

Puis enregistre dans sa table de routage (voir figure 3) $N_i$, $i-1$, et $K_i$.

Le nœud destinataire $n$ reçoit le message

$$< RREQ, trust, KP_{temp}, \{n, K_s, size(padding)\}_{KP_n}, padding,$$
$$\{0, KP_0, KP_{temp}, KS_{temp}, N_0, \{msg\}_{KS_0}\}_{K_s},$$
$$\{1, K_1, N_1, \{msg_1\}_{KS_1}\}_{KP_{temp}},$$
$$\dots$$
$$\{n-1, K_{n-1}, N_{n-1}, \{msg_{n-1}\}_{KS_{n-1}}\}_{KP_{temp}} >$$

En déchiffrant le troisième champ, il accède ainsi à la clé de session lui permettant d'accéder ensuite à la clé secrète $KS_{temp}$. Il peut maintenant prendre connaissance de l'identité de l'ensemble des nœuds présents sur le chemin ainsi que des clés de sessions pour chacun de ses nœuds. Il peut ainsi construire un oignon contenant le chemin de retour et un message de type $RREP$

$$< RREP, \{\dots\{\{\{N_1, K_1, \dots, N_{n-1}, K_{n-1}, K_n, size(padding), padding\}_{K_0},$$
$$N_0\}_{K_1}, N_1\}\dots\}_{K_{n-1}}, N_{n-1} >$$

Un nœud intermédiaire $i$ reçoit

$$< RREP, \{\dots\{\{\{N_1, K_1, \dots, N_{n-1}, K_{n-1}, K_n, size(padding), padding\}_{K_0},$$
$$N_0\}_{K_1}\}\dots\}_{K_i}, N_i >$$

Il utilise la table de routage et $N_i$ pour connaitre la clé $K_i$ à utiliser. Il déchiffre le message, enlève une couche et diffuse

$$< RREP, \{\dots\{\{\{N_1, K_1, \dots, N_{n-1}, K_{n-1}, K_n, size(padding), padding\}_{K_0},$$
$$N_0\}_{K_1}, N_1\}\dots\}_{K_{i-1}}, N_{i-1} >$$

Le nœud source reçoit

$$< RREP, \{N_1, K_1, \dots, N_{n-1}, K_{n-1}, K_n, size(padding), padding\}_{K_0}, N_0 >$$

Il déchiffre à l'aide de la clé $K_0$. Il obtient ainsi l'identité et les clés de chiffrement de l'ensemble des nœuds sur la route.

Pour la transmission de données, le nœud source utilise l'identité des nœuds et les clés pour construire un routage en oignon. Lorsqu'un nœud intermédiaire reçoit un message, il le déchiffre et diffuse le contenu déchiffré.

Table de routage :

| $N_i$ | $i-1$ | $K_i$ |
|---|---|---|

FIGURE 3 – Table de routage du protocole SDAR

**Propriétés de respect de la vie privée**

Le protocole SDAR protège correctement l'identité de la source et l'identité de la destination. En effet, l'identité de la source est chiffrée lors des échanges de messages *RREQ* et l'identité de la destination est protégée par l'utilisation d'une trappe.

Par contre, l'identité des nœuds intermédiaires n'est pas du tout assuré. Il est possible pour un attaquant collusif actif de connaitre l'identité des nœuds intermédiaires. Il suffit pour cela qu'il envoie un message d'un nœud qu'il contrôle à un autre nœud qu'il contrôle. Le destinataire connait en effet l'identité de tous les nœuds intermédiaires.

La protection de la position des nœuds du réseau n'est pas assuré. Un attaquant collusif actif peut connaitre la position de tous les nœuds présents sur le chemin et la position du destinataire. En effet, il lui suffit d'initier une découverte de route et à la réception de la réponse, il disposera de la liste ordonnée des nœuds et donc la position relative de chacun d'entre eux.

De plus, un attaquant global passif peut découvrir la position du nœud source et du nœud destinataire. Il suffit pour cela de regarder la taille des messages (que ce soit lors de la découverte de route ou lors de la transmission de données). Le padding utilisé ne permet pas de protéger des variations de tailles de messages.

En créant une route unique, SDAR ne respecte pas l'inassociabilité entre deux messages. De plus, tout comme les protocoles précédents, SDAR est aussi sensible à une attaque temporelle, permettant ainsi de relier un message à l'expéditeur ou au destinataire.

Ainsi, SDAR ne protège pas la vie privée des nœuds intermédiaires en ce qui concerne l'identité et de tous les nœuds concernant la position. Néanmoins, ce protocole est original de part l'utilisation d'un système de réputation. En supposant ce système bien conçu, un attaquant peut être limité dans ces attaques possibles et cela peut permettre d'exclure un nœud détecté comme malveillant. Cela n'est néanmoins pas suffisant pour faire de SDAR un protocole respectueux de la vie la vie privée.

**Performances**

L'utilisation du chiffrement asymétrique et du chiffrement symétrique impliquent des délais d'attente importants pour la négociation d'une route. De plus, comme un oignon est

utilisé pour les données, chaque nœud doit déchiffrer sa couche, le message diminuant en taille au fur et à mesure. La différence de taille entre le message original et le message dans l'oignon cryptographique peut être important, impactant ainsi sur la bande passante.

### 4.1.4 ARM

Le protocole ARM [18] repose sur l'utilisation d'un routage en oignon, de la notion de trappe et d'une méthode originale pour limiter la diffusion des messages en utilisant un TTL aléatoire. ARM utilise des valeurs aléatoires suivant une loi de distribution choisie par les auteurs pour initialiser et mettre à jour le TTL. La loi de distribution est choisie pour permettre à la valeur du TTL de diminuer.

Dans ARM, chaque nœud possède une identité permanente connue des autres nœuds. De plus le nœud source 0 et le nœud destinataire $n$ partagent une clé symétrique $K_{0,n}$ et un pseudonyme secret $Pseudo_{0,n}$. Chaque nœud du réseau définit avec ses voisins une clé de broadcast. Les auteurs proposent d'utiliser le *key management scheme* de Seys et Preneel [17].

**Description** Le nœud source envoie un paquet de type $RREQ$ pour initier la découverte de route.

$$< RREQ, Pseudo_{0,n}, ttl_0, KP_n, trapdoor_{dest}, \{N_0, K_0\}_{PK_n} >$$

$trapdoor_{dest}$ correspond à la trappe cryptographique. Elle est construite de la façon suivante

$$trapdoor_{dest} = \{n, K, KS_n\}_{K_{0,n}}, \{Pseudo_{0,n}\}_K$$

$KP_n$ et $KS_n$ sont respectivement une clé publique et une clé privée temporaires générées par le nœud source pour communiquer avec le nœud $n$, le nœud destinataire. $ttl_0$ correspond a un TTL (*Time To live*) du message choisi par le nœud 0. $N_0$ et $K_0$ sont des nombres aléatoires générés par le nœud source.

Lorsqu'un nœud intermédiaire $i$ reçoit le message suivant

$$< RREQ, Pseudo_{0,n}, ttl_{i-1}, KP_n, trapdoor_{dest}, \{\{\{N_0, K_0\}_{PK_n}, N_1, K_1\}_{PK_n} \dots N_{i-1}, K_{i-1}\}_{PK_n} >$$

il vérifie s'il est le destinataire du message. Pour cela, il vérifie si $Pseudo_{0,n}$ fait partie de sa liste de pseudonyme. Si c'est le cas, le nœud $i$ tente d'ouvrir la trappe à l'aide de la clé partagée. Si le test échoue, il vérifie dans un premier temps si $Pseudo_{0,n}$ est présent dans sa table de routage. Si c'est le cas, le nœud $i$ a alors déjà traité cette requête, il supprime donc le message. $i$ vérifie ensuite la valeur de $ttl_{i-1}$. Si celle-ci est supérieur à 1, alors le nœud $i$ mets à jour la valeur en tirant un nouveau nombre aléatoire suivant la loi de distribution et génère $N_i$ et $K_i$. Le nœud intermédiaire $i$ envoie ensuite le message suivant

$$< RREQ, Pseudo_{0,n}, ttl_i, KP_n, trapdoor_{dest},$$
$$\{\{\{\{N_0, K_0\}_{PK_n}, N_1, K_1\}_{PK_n} \dots N_{i-1}, K_{i-1}\}_{PK_n}, N_i, K_i\}_{PK_n} >$$

Enfin, il enregistre les informations suivantes dans sa table de routage (voir figure 4) $Pseudo_{0,n}, N_i, K_i, \{Pseudo_{0,n}\}_K$.

Le nœud destinataire reçoit le message suivant

$$< RREQ, Pseudo_{0,n}, ttl, KP_n, trapdoor_{dest}, \{\{\{N_0, K_0\}_{PK_n}, N_1, K_1\}_{PK_n} \dots N_{n-1}, K_{n-1}\}_{PK_n} >$$

Il effectue lui aussi la vérification du pseudonyme et déchiffre la trappe. Il transmet alors le message suivant

$$< RREQ, Pseudo_{0,n}, ttl, KP_n, trapdoor_{dest}, random >$$

Avec *random* un nombre aléatoire. La taille de *random* est choisi pour qu'il soit impossible de distinguer *random* du champ normalement émis si le nœud $n$ n'était pas le nœud destinataire.

A l'aide de la clé privée présente dans la trappe, le nœud destinataire accède à la liste des pseudonymes et des clés de chiffrement des nœuds présents sur la route. $n$ génère alors un message de type $RREP$, qu'il chiffre avec la clé de broadcast définie avec les nœuds voisins

$$< RREP, Pseudo_{0,n}, ttl, \{N_{n-1}, H(K_{n-2}), K, \dots \{N_0, K\}_{K_0}\}_{K_{n-1}}\}$$

Le nœud intermédiaire $i$ reçoit donc un message de la forme

$$< RREP, Pseudo_{0,n}, ttl, \{N_i, H(K_{i-1}), K, \dots \{N_0, K\}_{K_0}\}_{K_i}\}$$

Il vérifie dans un premier temps s'il a déjà reçu une réponse en utilisant $Pseudo_{0,n}$ comme numéro de séquence unique. S'il a déjà traité le message ou qu'il n'a pas reçu le message $RREQ$ correspondant, il envoie le message suivant

$$< RREP, Pseudo_{0,n}, ttl, random >$$

Sinon, il déchiffre la première couche de l'oignon en vérifiant que le premier champ correspond bien à $N_i$. Il vérifie ensuite que le message provient bien du nœud destinataire en déchiffrant à l'aide de $K$ la valeur $\{Pseudo_{0,n}\}_K$ et en la comparant avec le pseudonyme présent dans le message. Si les deux valeurs sont égales, alors le nœud destinataire connait bien la clé $K$ et a donc bien déchiffré la trappe. Enfin, $i$ transmet le message suivant

$$< RREP, Pseudo_{0,n}, ttl, \{N_{i-1}, H(K_{i-2}), K, \dots \{N_0, K\}_{K_0}\}_{K_{i-1}}\}$$

et stocke dans une autre table de routage les informations nécessaires (voir figure 4). $H(K_{i-1})$ correspond à un secret que le nœud $i$ partage avec le nœud $i-1$ et $H(K_i)$

correspond à un secret que le nœud $i$ partage avec le nœud $i+1$.

Le nœud source reçoit le message $RREP$ suivant

$$< RREP, Pseudo_{0,n}, ttl, \{N_0, K\}_{K_0} >$$

Il vérifie que $K$ est la même clé que lors de l'envoi du message $RREQ$. Cela lui sert de preuve de l'identité de la personne ayant répondu à sa requête.

La transmission de messages consiste à utiliser les secrets partagés et un compteur comme identifiant de messages, tout en chiffrant le contenu à l'aide des secrets partagés, toutes les informations étant présentes dans une des tables de routage.

| Table de routage : | $Pseudo_{0,n}$ | $N_i$ | $K_i$ | $\{Pseudo_{0,n}\}_K$ |
|---|---|---|---|---|

FIGURE 4 – Table de routage du protocole ARM

**Propriétés de respects de la vie privée**

Le protocole ARM respecte l'identité de tous les nœuds du réseau. la protection de l'identité de la source est assurée par un pseudonyme partagé par les deux nœuds. Quant à la destination, l'utilisation d'une trappe permet de protéger son identité. Enfin, pour les nœuds intermédiaires, une clé et un nombre aléatoire (qui sont renouvelé à chaque nouvelle requête) remplace leurs identités.

Le respect de la vie privée concernant la position des nœuds n'est pas assuré par le protocole. Un attaquant collusif actif peut déterminer la position relative des nœuds présent sur le chemin. Cela consiste à envoyer une demande de requête à partir d'un nœud $a$ que l'attaquant contrôle à destination d'un nœud $b$ que l'attaquant contrôle également. $n$ connait ainsi la liste des pseudonymes et des clés pour chaque nœud sur la route. Or cette liste doit forcement être ordonné. Il est ainsi possible pour un attaquant de connaitre la position relative des nœuds intermédiaires.

Un attaquant global passif peut suivre les différents messages $RREQ$ et $RREP$ en utilisant comme identifiant le champ $Pseudo_{0,n}$. Cela permet ainsi de retrouver l'intégralité de la route et ainsi de trouver la position relative des différents nœuds (source, destination et intermédiaires).

L'inassociabilité n'est pas non plus assurée par ARM. De part la création d'une route unique, deux messages utilisant cette route proviennent nécessairement d'une communication entre le nœud source et le nœud destinataire.

De plus, un attaquant global passif peut aussi utiliser une attaque temporelle (déjà vue dans les protocoles précédents) permettant de retrouver la source et la destination d'un message.

L'utilisation d'un padding et d'un ttl aléatoire est une technique permettant effectivement d'empêcher les nœuds intermédiaires de connaitre la position de la source. Cela n'est néanmoins pas suffisant, d'autres attaques sont possibles contre le protocole ARM. De plus, le choix de la loi de distribution concernant le padding et le ttl n'est pas suffisamment étudié par les auteurs. Il est ainsi difficile de se convaincre du bon fonctionnement de cette solution.

**Performances**

ARM utilise le chiffrement au niveau local (diffusion en broadcast) et un chiffrement pour les informations de routage. Le chiffrement au niveau local implique un déchiffrement / chiffrement pour chaque nœud. De plus, les auteurs n'ont pas précisés quels algorithmes étaient utilisés, rendant ainsi difficile l'évaluation de l'impact du chiffrement sur les performances du protocole.

### 4.1.5 MASK

Le protocole MASK [22] utilise deux concepts pour protéger la vie privée : un MIX-net permettant de répartir les messages sur plusieurs chemins et d'authentification mutuelle permettant de générer des pseudonymes et des clés partagées entre deux nœuds. Les pseudonymes servent à identifier un lien entre deux nœuds.

Un MIX-net est un réseau utilisant une chaine de serveurs pour transmettre le message. Le routage en oignon est un exemple de MIX-net. MASK permet la création de plusieurs chemins pour une même destination. Ces chemins permettent la construction d'un MIX-net et permet ainsi de répartir aléatoirement les messages sur les différents chemins possibles.

L'authentification mutuelle entre deux nœuds prend en compte l'existence de plusieurs MANETs déconnectés les uns des autres. Afin d'éviter toute fuite d'information, un nœud ne peut déterminer de quel groupe est un autre nœud (nonobstant l'appartenance des deux nœuds au même groupe). Pour réaliser cela, les auteurs utilisent du chiffrement basé sur les paires.

Ainsi, si les deux nœuds font partie du même groupe, alors après l'authentification mutuelle, les deux nœuds partagent la même clé. Cette clé partagée permet de dériver une clé de session $K^l_{1,2}$ et un identifiant du lien $L^l_{1,2}$. $l$ représente le numéro de la clé ou le numéro de l'identifiant. Ainsi, $L^l_{1,2}$ représente le $l^{eme}$ lien entre le nœud 1 et le nœud 2. En se synchronisant, les nœuds peuvent ainsi changer régulièrement de clé et d'identifiant.

Le fonctionnement de l'algorithme de routage repose sur la pseudonymisation des liens et sur le principe de MIX-net. Trois tables sont utilisées par le protocole de routage : *Forwarding Route Table* (FRT), *Reverse Route Table* (RRT) et *Target LinkID Table* (TLT) (voir figure 5).

Les auteurs supposent l'existence d'une autorité de confiance est nécessaire avant la création du MANET. Cette autorité détermine les paramètres publics nécessaire pour le fonctionnement de MASK.

**Déroulement du protocole**

Le nœud source 0 envoie un message $RREQ$

$$< RREQ, id_{RREQ}, n, seq_{dest}, P_0 >$$

$id_{RREQ}$ est l'identifiant unique de la requête. $n$ correspond à l'identité réelle de la destination. $seq_{dest}$ est un numéro de séquence propre à la destination. $P_0$ est un pseudonyme temporaire du nœud source.

Un nœud intermédiaire $i$ reçoit le message $RREQ$ suivant

$$< RREQ, id_{RREQ}, n, seq_{dest}, P_{i-1} >$$

Il vérifie dans un premier temps s'il est le nœud destinataire. Si ce n'est pas le cas, il regarde s'il ne connait pas déjà une route pour cette destination. Si c'est le cas, il envoie un message $RREP$ de la forme

$$< L^{\alpha}_{i-1,i}, \{RREP, n, seq_{dest} + 1\}_{K^{alpha}_{i-1,i}} >$$

en utilisant l'identifiant de lien généré lors de l'authentification mutuelle entre lui et le nœud $i-1$. Enfin, s'il a déjà traité la requête, il supprime le message. Sinon, il enregistre dans sa RRT $n$, $seq_{dest}$ et $P_{i-1}$. Il diffuse ensuite le message suivant

$$< RREQ, id_{RREQ}, n, seq_{dest}, P_i >$$

Le destinataire reçoit

$$< RREQ, id_{RREQ}, n, seq_{dest}, P_{n-1} >$$

Le nœud $n$ diffuse le message reçu en remplaçant le pseudonyme par le sien. Il peut ensuite, après la phase d'authentification mutuelle lui permettant de récupérer $L^{\beta}_{n,n-1}$, envoyer le message suivant

$$< L^{\beta}_{n-1,n}, \{RREP, n, seq_{dest} + 1\}_{K^{beta}_{n-1,n}} >$$

à destination du nœud dont le pseudonyme est $P_{n-1}$. Il enregistre ensuite $L_{n-1,n}^{\beta+1}$ dans sa TLT. Cette table permet au nœud de distinguer rapidement que le prochain message envoyé par le nœud $n-1$ est pour lui (le nœud $n$).

Le nœud intermédiaire $i$ voit arriver le message de type $RREP$ suivant

$$< L_{i+1,i}^{\gamma}, \{RREP, n, seq_{dest}+1\}_{K_{i+1,i}^{\gamma}} >$$

A partir de l'identifiant du lien, il détermine la clé à utiliser pour déchiffrer le message. En recherchant l'identifiant de la destination et le numéro de séquence dans sa RRT, le nœud accède au pseudonyme $P_{i-1}$ du nœud lui ayant envoyé le message. Ce pseudonyme permet de déterminer, à l'aide de l'authentification mutuelle, l'identifiant du lien et la clé à utiliser. $i$ observe ensuite $seq_{dest}$ afin de déterminer s'il faut enregistrer une nouvelle entrée (pas d'entrée déjà existante ou numéro de séquence égal) ou mettre à jour une entrée existante (si le numéro de séquence est plus récent dans le message reçu) FRT. Une entrée dans la FRT se présente de la façon suivante : $n$, $seq_{dest}$, $L_{i-1,i}^{\alpha}$ et $L_{i,i+1}^{\gamma}$ (respectivement l'identifiant du lien entre le nœud $i-1$ et $i$ et l'identifiant du lien entre le nœud $i$ et $i+1$). Le nœud intermédiaire peut ensuite envoyer le message suivant

$$< L_{i-1,i}^{\alpha}, \{RREP, n, seq_{dest}+1\}_{K_{i-1,i}^{alpha}} >$$

Le nœud source reçoit le message suivant

$$< L_{1,0}^{\delta}, \{RREP, n, seq_{dest}+1\}_{K_{1,0}^{\delta}} >$$

et réalise le même traitement que lorsqu'un nœud intermédiaire reçoit un message de type $RREP$. Mais puisque sa RRT est vide, il enregistre dans sa FRT les informations suivantes : $n$, $seq_{dest}$, $null$, $L_{1,0}^{\delta}$.

Après la phase de la découverte de route, une ou plusieurs routes totalement disjointes existent entre le nœud source et le nœud destination. Tous les nœuds intermédiaires disposent grâce à la FRT de l'identifiant du lien précédent et de l'identifiant du lien suivant. Un nœud peut ainsi simplement remplacer l'identifiant et diffuser le message. Il est aussi possible d'utiliser la clé liée à l'identifiant de lien pour chiffrer les données. Un message transportant les données ressemblera alors à

$$< L_{i-1,i}^{\epsilon}, \{data\}_{K_{i-1,i}^{\epsilon}} >$$

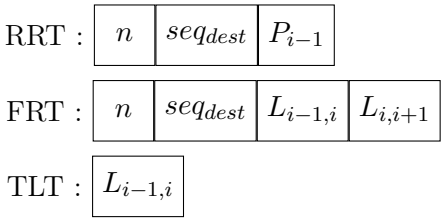


FIGURE 5 – Tables de routage du protocole MASK

**Propriétés de respect de la vie privée**

MASK assure, en utilisant des pseudonymes temporaires, la protection de l'identité de la source et des nœuds intermédiaires. Par contre, N'importe quel nœud du réseau, qu'il soit attaquant ou non, peut connaitre l'identité du destinataire. En effet, celle-ci est diffusée en clair sur le réseau. MASK n'assure donc pas la protection de l'identité de la destination.

Un attaquant global passif peut déterminer la position de la source et de la destination. En effet, à part dans la phase *RREQ*, le nœud source et le nœud destination ne diffusent pas de messages lorsque celui-ci leur est destiné. Les auteurs proposent ainsi un système de diffusion des messages au delà des nœuds sources et destination. Cela n'est néanmoins pas suffisant. Un attaquant global passif peut écouter les messages échangés sur le réseau. En observant les points de convergences, un tel attaquant peut retrouver la source et la destination du message. En effet, les seuls nœuds étant sur l'ensemble des chemins totalement disjoints sont le nœud source et le nœud destination. Ainsi, en retraçant les messages émis par chaque nœud, il est possible de trouver ces points de convergence. Ainsi, le protocole ne permet pas de protéger la position de la source et de la destination.

En ce qui concerne les nœuds intermédiaires, il n'existe pas d'attaques connues permettant de déterminer leurs positions. En effet, dans les protocoles précédents, l'attaque consistait à utiliser soit la taille des messages soit un TTL pour déterminer la position relative d'un nœud. Dans MASK, les messages sont de taille fixe et il n'utilise pas de TTL ou un champ équivalent.

Un attaquant collusif local ne peut pas déterminer si deux messages sont liés entre eux. Cette propriété très intéressante d'un point de vue anonymat est assuré par le MIX-net (le nœud source choisissant aléatoirement le prochain saut dans sa liste des sauts possibles). Un tel attaquant peut tout au plus déterminer une partie des chemins en réunissant les différentes RRT. Dans le pire des cas, l'attaquant peut avoir compromis l'ensemble des nœuds d'une route. Il peut ainsi observer le trafic échangé. Cependant, le système de MIX-net ne lui permet pas de récupérer l'ensemble du trafic entre le nœud source et le nœud

destination. Ainsi, le protocole assure l'inassociabilité entre deux messages.

L'inassociabilité entre l'expéditeur et le message et l'inassociabilité entre le destinataire et le message sont assurées par la réutilisation d'un même pseudonyme temporaire pendant une durée de temps. Ainsi, deux nœuds déjà relié et possédant un identifiant de lien peuvent le réutiliser. Si les deux nœuds sont sur un chemin concernant deux communications différentes, il n'est pas possible de relier le message à une communication particulière et donc à un expéditeur ou à un destinataire.

#### Performances

Les auteurs de MASK utilisent RC6 comme algorithme de chiffrement symétrique. Le chiffrement implique un temps de latence pour chaque nœud, temps nécessaire pour le chiffrement / déchiffrement. Cependant, l'utilisation du chiffrement symétrique permet de limiter le temps de latence. Une des forces de ce protocole est qu'il est résistant à la forte mobilité des nœuds et à la congestion réseau. Cela repose sur l'utilisation de MIX-net. Ainsi, pour une destination choisie, plusieurs circuits sont possibles. Si un nœud participant au premier circuit devient indisponible, il est possible très rapidement d'utiliser le deuxième circuit. Ce choix aléatoire permet de diluer le trafic sur une partie du réseau, et non pas sur un seul chemin.

### 4.2 Protocoles basés sur la position

Contrairement aux algorithmes précédents, ALARM et PRISM sont des protocoles ad hoc basés sur la position visant à faire respecter la vie privée.

#### 4.2.1 ALARM

Le principe de fonctionnement d'ALARM [7] repose sur la propagation de l'information concernant la position de chaque nœud dans l'ensemble du réseau. La position exacte du destinataire est protégée en utilisant du chiffrement asymétrique.

Afin de protéger l'identité de la source, la signature de groupe est utilisée. La signature de groupe permet aux nœuds de ce groupe de signer n'importe quel message avec la clé privée du groupe. Cette signature peut être vérifiée par n'importe qui possédant la clé publique correspondante. De plus, il n'est pas possible à un nœud de déterminer quel nœud à signé le message. La signature de groupe nécessite une étape hors ligne d'initialisation du groupe, initialisation réalisée par le superviseur du groupe (*Group Master*).

Pour le bon fonctionnement du protocole, les auteurs supposent que chaque nœud connaît sa position (en pratique, il suffit d'utiliser un récepteur GPS). Le protocole nécessite aussi une horloge synchronisée. Cette synchronisation n'a pas besoin d'être une synchronisation forte. A nouveau, un récepteur GPS permet de répondre à ce problème. La

dernière hypothèse est une hypothèse très forte : dans une période de temps fixée, au moins K nœuds doivent bouger.

**Déroulement du protocole**

Le protocole découpe le temps en créneaux d'une durée $T$. A chaque début de créneau, chaque nœud diffuse à tous le monde un message LAM contenant les informations suivantes : la position (GPS), un timestamp, une clé publique temporaire et la signature de groupe des précédents champs

$$< Position, Timestamp, KP_i, group\_signature(Position, Timestamp, KP_i >$$

Lorsqu'un nœud reçoit un message LAM, il vérifie dans un premier temps la signature de groupe. Si la signature est invalide, il détruit le message. Sinon, le nœud transmet ce message LAM.

En collectant tous les messages LAM, un nœud peut construire une carte positionnant l'ensemble des nœuds du réseau.

Lorsque le nœud source souhaite envoyer des données à une zone géographique, il utilise sa connaissance de la position des nœuds dans le réseau pour déterminer les nœuds concernés par son message. Il envoie ensuite à chacun des nœuds le message suivant (ici, pour le nœud $n$)

$$< group\_signature(Position, Timestamp, KP_n), \{data\}_{KP_n} >$$

Le premier champ correspond à la signature de groupe réalisée par le nœud destinataire. Ce champ sert d'identifiant de la destination. Le nœud source chiffre ensuite à l'aide de la clé publique temporaire du nœud $n$ ($KP_n$) le champ *data* contenant les informations qu'il souhaite diffuser.

Un nœud intermédiaire $i$ reçoit le message suivant

$$< group\_signature(Position, Timestamp, KP_n), \{data\}_{KP_n} >$$

Il vérifie si il est le destinataire du message en comparant la signature de groupe présent dans le message avec la signature de groupe qu'il a envoyé dans le message LAM. Si il n'est pas le destinataire, il diffuse le message. Sinon, il déchiffre à l'aide de $KS_n$ le champ *data*.

Nous pouvons remarquer qu'il n'existe pas de route pour atteindre la destination, chaque nœud diffuse le message. Ce type de routage est appelé routage par inondation.

**Propriétés de respect de la vie privée**

ALARM étant un protocole basé sur la position, les propriétés de protection de l'identité n'a pas de sens pour ce protocole.

Le fonctionnement d'ALARM repose sur la diffusion dans tout le réseau de la position de l'ensemble des nœuds. Ainsi, un attaquant global passif peut reconstruire le graphe du réseau.

La signature de groupe permet d'assurer l'inassociabilité entre l'expéditeur et le message. En effet, il est impossible pour un nœud du réseau de distinguer la signature provenant d'un nœud d'une autre signature provenant d'un autre nœud dans un message LAM. Pour la diffusion de données, le nœud expéditeur ne fait qu'envoyer un message de type "tire et oublie". Si il veut avoir une réponse, il doit utiliser le champ *data*. Un message de données ne contenant alors aucun lien avec l'expéditeur, l'inassociabilité de entre l'expéditeur et un message est assurée.

L'inassociabilité entre les messages est assurée. En effet, comme il n'est pas possible de repérer qui est l'expéditeur d'un message, deux messages peuvent provenir de deux nœuds différents pour une même destination.

ALARM n'assure pas l'inassociabilité entre le destinataire et un message. En effet, la signature de groupe que le nœud destinataire utilise dans le message LAM est utilisé comme identifiant pour les messages contenant des données. Il est ainsi très facile de savoir à qui un nœud envoie des informations.

**Performances**

Les performances d'ALARM sont difficiles à évaluer. En effet, dans les tests réalisés par les auteurs, seul le respect de la vie privée par rapport à la vitesse des nœuds est analysé. Ainsi, il n'y a pas d'informations particulières sur le temps de latence pour la diffusion d'un message, la surcharge du réseau, la charge pour chaque nœud... De plus, ALARM est un des rares protocoles basé sur la position, rendant la comparaison avec d'autres protocoles difficile.

Néanmoins, l'utilisation de l'inondation pour transmettre un message est extrêmement inefficace. Pour chaque message à diffuser, tous les nœuds doivent participer. De plus, ALARM n'utilise pas la position des différents nœuds pour router intelligemment.

### 4.2.2 PRISM

Tout comme ALARM, PRISM [8] est un protocole basé sur la position. Ce protocole utilise lui aussi la signature de groupe vu précédemment. Contrairement à ALARM, PRISM

créé véritablement une route pour atteindre la destination depuis le nœud source.

Les deux protocoles étant très proche, le protocole PRISM possède les mêmes hypothèses qu'ALARM. Ainsi, un nœud doit connaitre sa position et avoir une horloge synchronisée. De plus, PRISM considère le réseau comme n'étant pas de confiance.

**Déroulement du protocole**

Le nœud 0 initie une découverte de route vers une zone en envoyant un message de type $RREQ$

$$< RREQ, dest, PK_0, timestamp_0, group\_signature(RREQ) >$$

$dest$ contient la zone destinatrice du message, $PK_0$ une clé publique temporaire, $timestamp_O$ un timestamp et $group_signature(RREQ)$ est la signature de groupe de tous les autres champs.

Un nœud intermédiaire $i$ reçoit le même message que celui envoyé par le nœud source. Le nœud vérifie tout d'abord si $timestamp_0$ est valide. Dans le cas contraire, le message est supprimé. Sinon, un hash du message (noté par la suite $h(RREQ)$) est réalisé et permet de déterminer si le message à déjà été traité. Si le message a déjà été traité, il est supprimé. Sinon, le nœud $i$ vérifie si il se trouve dans la zone destinatrice $dest$. Si le nœud n'est pas dans la zone, il enregistre $h(RREQ)$ dans un cache et retransmet le message à l'identique.

Lorsque le nœud destinataire reçoit le message $RREQ$, il vérifie tout d'abord qu'il est bien présent dans la zone destinatrice puis vérifie la signature de groupe $group\_signature_0$. Si la signature est correcte, il envoie un message de réponse de type $RREP$

$$< RREP, h(RREQ), \{K_s, location\}_{PK_0}, group\_signature(RREP) >$$

$K_s$ est une clé de session générée par le nœud $n$, $location$ la position exacte du nœud destinataire et $group\_signature(RREP)$ est la signature de tous les autres champs du message. La clé de session et la position du nœud sont chiffré à l'aide de la clé publique présente dans le message $RREQ$.

Table de routage : | $h(RREQ)$ | $h(RREP)$ |

Cache : | $h(RREQ)$ |

FIGURE 6 – Tables de routage de PRISM

Le nœud intermédiaire $i$ reçoit le même message $RREP$ que celui envoyé par la destination. Lorsqu'il reçoit le message, il vérifie dans un premier temps si $h(RREQ)$ est présent dans son cache. Si oui, cela signifie qu'il est sur la route pour rejoindre le nœud source. Il enregistre donc dans une table de routage (voir figure 6) $h(RREQ), h(RREP)$ ainsi que l'heure de la création de la route. Si $h(RREQ)$ n'est pas présent dans le cache, il supprime le message.

Lorsque le nœud source reçoit un message $RREP$, il effectue d'abord les mêmes vérifications que le nœud intermédiaire. il vérifie ensuite la signature de groupe. Il déchiffre la clé de session $K_s$ et la localisation du nœud destinataire *location*.

Une fois la route créée, chaque message envoyé de la source vers la destination aura pour identifiant unique $< h(RREQ), h(RREP) >$ et réciproquement, chaque message envoyé de la destination vers la source aura pour identifiant unique $< h(RREP), h(RREQ) >$.

**Propriétés de respect de la vie privée**

Tout comme ALARM, les propriétés de respect de la vie privée concernant l'identité n'est pas cohérente avec un protocole basé sur la position.

PRISM protège la position des nœuds intermédiaires et du nœud destinataire. En effet, les nœuds intermédiaires ne modifiant pas les messages transmis, aucune information ne peut être utilisé pour retrouver leurs positions. Pour le nœud destinataire, sa position, envoyé au nœud source, est chiffré. Ce dernier point est à nuancer. Un nœud destinataire se trouve par définition dans la zone de destination. Un attaquant peut ainsi avoir une information peu précise sur la localisation de la destination.

Cependant, un attaquant global passif peut connaitre la position du nœud source. En effet, le timestamp est fourni lors de l'envoi du message $RREQ$. Cette indication de temps, couplé à la vitesse de propagation des ondes, permet de déterminer la position relative de la source.

PRISM ne respecte pas l'inassociabilité d'un expéditeur avec un message, d'un destinataire avec un message et d'un message avec un message. En effet, ce protocole est sensible à l'attaque temporelle déjà présentée. De plus, comme une seule route est créée et que les identifiants changent à chaque nouvelle demande de route, un message utilisant ce chemin est donc synonyme de communication entre le nœud source et le nœud destination.

**Performances**

L'analyse des performances de PRISM est difficile. En effet les auteurs n'ont pas analysé les performances de PRISM d'un point de vue transmission de messages, bande passante

utilisée... mais du point de vue du respect de la vie privée.

De manière générale, nous pouvons néanmoins dire que PRISM est plus performant qu'ALARM. En effet, la propagation d'une suite de message ne se fait pas par inondation, mais par la création d'une route. Cela implique une utilisation moins importante de la bande passante, un nombre de nœuds participant au routage moins important et un meilleur anonymat (en effet, un attaquant local ne se trouvant pas sur la route n'a pas connaissance des messages transmis entre le nœud source et le nœud destinataire).

### 4.3 Conclusion

L'ensemble des protocoles étudiés vise à protéger la vie privée. La figure 7 propose sous forme de tableau l'ensemble des propriétés que doivent assurer les nœuds. Malheureusement, aucun des protocoles ad hoc étudiés ne permet d'assurer un respect de la vie privée à l'ensemble des nœuds du réseau. Le protocole MASK est le seul à assurer l'inassociabilité entre les messages, entre le message et la source et enfin entre le message et la destination. Malheureusement, l'identité du nœud destinataire n'est pas protégée par MASK, ainsi que les informations concernant la position de la source et de la destination.

## 5 Proposition de protocole : NoName

Comme nous l'avons vu dans la section 3.2, un protocole de routage ad hoc respectueux de la vie privée doit prendre en compte : l'anonymat de l'identité de tous les nœuds, l'anonymat de la position de tous les nœuds et l'inassociabilité entre la source et le message, entre le message et la destination et entre deux messages provenant de la source et allant vers la destination.

Bien qu'étant très intéressant, MASK ne respecte pas ces propriétés. En effet, l'identité du nœud destinataire apparait en clair dans le message de requête de route. De plus, le routage à plusieurs chemins que propose MASK pose quelques problèmes. Tout d'abord, il existe deux points de convergence dans le routage à plusieurs chemins proposé par MASK : la source et la destination. Une écoute passive peut ainsi révéler la position de la source et de la destination. De plus, le nombre de routes dépend du nombre de voisins du destinataire recevant le message de requête. Cela implique une efficacité moindre et *de facto* une diminution de l'anonymat dans certains réseaux (typiquement les réseaux à peu de nœuds).

Le protocole proposé, NoName, se base sur le fonctionnement de MASK tout en visant à respecter ces propriétés.

| | ANODR | ASR | SDAR | ARM | MASK | ALARM | PRISM |
|---|---|---|---|---|---|---|---|
| identité de la source | √ | √ | √ | √ | √ | - | - |
| identité de la destination | √ | √ | √ | √ | × | - | - |
| identité des nœuds intermédiaires | √ | √ | × | √ | √ | - | - |
| position de la source | × | × | × | × | × | × | × |
| position de la destination | × | × | × | × | × | × | √ |
| position des nœuds intermédiaires | × | × | × | × | √ | × | √ |
| inassociabilité émetteur - message | × | × | × | × | √ | √ | × |
| inassociabilité destinataire - message | × | × | × | × | √ | × | × |
| inassociabilité message - message | × | × | × | × | √ | × | × |

FIGURE 7 – Tableau récapitulatif du respect de la vie privée des protocoles analysés

## 5.1 Concepts

### 5.1.1 Trappe

Une trappe est un système permettant de cacher l'identité réelle du destinataire. Plus formellement, c'est une fonction à sens unique, que seul le destinataire, ayant une information supplémentaire, peut inverser. "L'ouverture" d'une trappe consiste à inverser la fonction. Une implémentation possible d'une trappe consiste à utiliser le chiffrement asymétrique.

Dans NoName, chaque nœud possède une clé privée ($S_i$) et une clé publique ($P_i$). La construction d'une trappe pour une destination donnée consiste à chiffrer des informations avec la clé publique du destinataire, $P_{dest}$.

$$Trapdoor = \{id_{dest}||ID_{REQ}||Nounce\}_{P_{dest}}$$

$id_{dest}$ correspond à l'identité réelle du destinataire, $ID_{REQ}$ à l'identifiant unique d'un message de type demande de route et $Nounce$ à un nombre aléatoire généré par le nœud source. Ce nombre permet à la destination de prouver l'ouverture correcte de la trappe.

### 5.1.2 Routage à plusieurs chemins

Le routage à plusieurs chemins consiste à créer plusieurs routes pour une même destination. Cela permet une répartition du trafic entre les différents chemins et une plus grande résistance à la disparition d'une route.

Lou *et al.* [13] distinguent deux familles de chemins : les chemins totalement disjoints et les chemins partiellement disjoints.

Les chemins totalement disjoints ont une intersection vide, nonobstant le nœud source et le nœud destinataire. Dans [13], les auteurs définissent SPREAD, un protocole utilisant des chemins totalement disjoints. SPREAD utilise un système de secret réparti : un message est décomposé en plusieurs morceaux puis est ensuite envoyé à travers différentes routes. Les chemins totalement disjoints permettent ici de mieux se protéger de la compromission d'un nœud. En effet, avec SPREAD, un nœud compromis ne verra qu'un morceau du message. Un autre exemple de protocole de routage utilisant du routage à plusieurs chemins totalement disjoints est MASK.

Les chemins partiellement disjoints peuvent avoir un ou plusieurs nœuds intermédiaires en commun. Ce système à l'avantage de permettre plus de chemins que dans la famille précédente. Il est ainsi plus résistant à la mobilité et ne permet pas de distinguer le nœud source et le nœud destinataire. En effet, au contraire des chemins totalement disjoints, l'intersection de plusieurs chemins ne désigne pas formellement un nœud comme étant la source ou la destination. Cette famille est principalement utilisée lors d'utilisation de chemins de secours (c'est le cas par exemple dans APR [14] ou bien AODV-BR [12]).

Un routage à plusieurs chemins partiellement disjoints semble donc le plus intéressant pour respecter la vie privée. Malheureusement, la création de ces routes sans la présence

de boucle n'est pas trivial. En effet, un nœud doit pouvoir prendre en compte une même requête envoyée par différents nœuds.

Il existe deux méthodes usuelles permettant de détecter et donc de se protéger de boucles dans un chemin : l'utilisation d'un TTL (Time To Live, utilisé par exemple par le protocole IP) et l'utilisation d'une liste contenant les nœuds traversés.

Le TTL est aisé à mettre en œuvre : un compteur est initialisé par l'expéditeur du message. Chaque nœud traitant le message décrémente de un le compteur. Si le compteur arrive à zéro, alors le message a traversé un trop grand nombre de nœuds (cela peut impliquer la présence d'une boucle dans le routage). Malheureusement, suivant l'initialisation du compteur, la détection peut être très longue. L'utilisation d'un TTL donne également une indication de la distance du nœud source. La valeur initiale étant connue, il est possible pour un nœud intermédiaire de déterminer le nombre de sauts (et donc une position relative) pour atteindre le nœud source. ARM utilise un champs TTL aléatoire, permettant justement de contrer ce type d'attaque. Malheureusement, l'efficacité de cette solution n'a pas été démontré par les auteurs.

Il est possible de stocker dans une liste l'identité des nœuds traversés par la requête. Afin de respecter l'anonymat des nœuds, il est possible de remplacer l'identité du nœud par un nombre généré aléatoirement. Cependant, la liste grossit au fur et à mesure, permettant ainsi d'avoir une information indirecte de la position relative de la source.

### 5.1.3 Filtre de Bloom

Le filtre de Bloom est une structure de données probabiliste permettant de stocker un ensemble de valeurs. Cette structure a été décrit pour la première fois dans [1]. Elle est très utilisée en informatique dans le domaine des bases de données. Dans [4], les auteurs décrivent les différentes utilisations possibles d'un filtre de Bloom.

Un filtre de Bloom ($FB$) est représenté par un tableau de $m$ bits. Lors de la construction du filtre, $m$ est défini ainsi que $k$ fonctions de hash (notée $H_i, i \in \{0..k\}$). Deux opérations sont possibles sur cette structure de données : l'ajout d'un élément et la vérification de la présence d'un élément dans l'ensemble.

- Ajout d'un élément $Elt$ au filtre de Bloom $FB$ :

  $$\forall i \in \{0..k\}, FB(H_i(Elt)) = 1$$

  Le principe est le suivant : le hash d'un élément va correspondre à un indice dans le tableau de bits. Pour chaque fonction de hash, on calcule l'indice puis le bit situé à l'indice dans le tableau est mis à 1.
- Recherche d'un élément $Elt$ dans le filtre de Bloom $FB$ :

  $$\forall i \in \{0..k\}, FB(H_i(Elt)) == 1 \Rightarrow Elt \in FB$$

  La recherche d'un élément est très similaire à l'ajout d'un élément. On vérifie pour chaque indice calculé si le bit situé dans le tableau à cet indice est à 1.

Par construction, le filtre de Bloom est sensible aux faux positifs (*ie*, un élément est détecté comme faisant partie du filtre de Bloom sans y être). En effet, il est possible qu'un élément non présent dans l'ensemble soit détecté. Soit $X$ l'élément non présent et $Y_{0..j}$ les éléments déjà présent dans l'ensemble. Pour que $X \in FB$, il suffit que pour chaque $H_i(X)$, il existe $H_a(Y_b), a \in \{0..k\}, b \in \{0..j\}$ tel que $H_i(X) == H_a(Y_b)$. Il est possible d'estimer la probabilité de faux positifs. Ainsi suivant [2], la probabilité est de :

$$p_{k,n,m} = \frac{1}{m^{k(n+1)}} \sum_{i=1}^{m} \binom{m}{i} \left\{ {kn \atop i} \right\}$$

Avec $k$ le nombre de fonctions de hash, $n$ le nombre d'éléments de l'ensemble et $m$ la taille du filtre de Bloom.

Une propriété intéressante de ce genre de filtre est l'absence de faux négatifs (*ie*, un élément est détecté comme ne faisant pas partie du filtre de Bloom en y étant). Autrement dit, si la recherche d'élément donne une réponse négative, alors l'élément n'est définitivement pas dans l'ensemble.

Un filtre de Bloom peut être utiliser pour conserver l'historique des nœuds traversés par un message, en stockant le filtre de Bloom directement dans le message. La faible taille de cette structure permet ainsi de limiter l'impact du protocole de routage sur l'autonomie des nœuds. L'absence de faux négatif permet de s'assurer la non existence de boucle de routage. Cette structure à déjà été utilisé dans [21] afin de remplacer les TTL. Néanmoins, à notre connaissance le filtre de Bloom n'a pas été utilisé pour construire de multiple chemins. Il a néanmoins été utilisé dans un protocole de réseau ad hoc, ODAR [20].

Les faux positifs ne font que limiter les routes possibles. En effet, si un nœud se détecte par erreur dans le filtre de Bloom, il va simplement ne pas prendre en compte ce message et ainsi ne pas participer à la création d'une route.

## 5.2 Fonctionnement & algorithme

NoName est un protocole réactif s'inspirant du fonctionnement du protocole MASK. La création de route repose sur deux types de messages : les messages de type *RREQ* (*Route Request*) et *RREP* (*Route Response*). La source envoie un message de type *RREQ* lorsqu'elle ne possède pas de route pour une destination donnée. Ce message sera diffusé dans tout le réseau. Le nœud destinataire, lorsqu'il reçoit un message de type *RREQ*, répond en envoyant un message de type *RREP*. C'est ce second message qui est responsable de la création de la route pour atteindre la destination.

NoName repose sur l'utilisation d'un filtre de Bloom pour permettre la création de chemins partiellement disjoints sans transmettre l'identité ou la position des nœuds. Pour préserver l'identité des nœuds, des pseudonymes temporaires sont utilisés ainsi que la notion de trappe pour assurer l'anonymat de la destination.

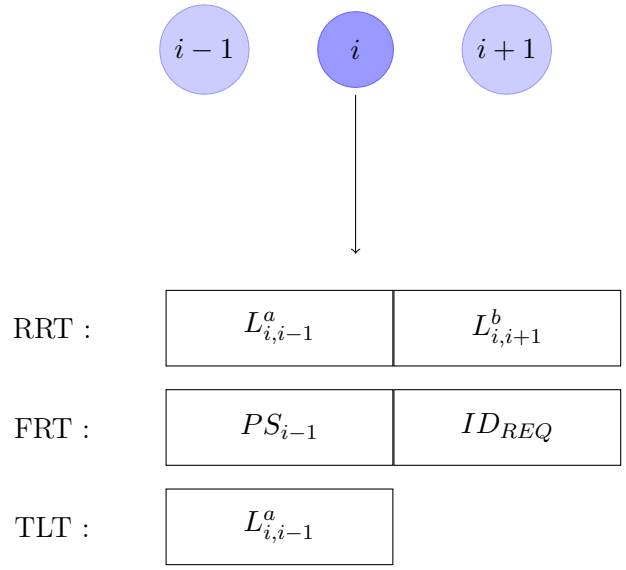


FIGURE 8 – structures des tables maintenues par un nœud

Chaque nœud maintient trois tables : la FRT (*Forwarding Route Table*), la RRT (*Reverse Route Table*) et TLT (*Target LinkID Table*). La figure 8 décrit la structure de ces différentes tables. La FRT est une table temporaire servant à stocker les informations concernant une demande de route. La RRT est utilisée lors de l'envoi de données : elle permet de lier deux liens entre eux. La TLT permet à un nœud de se reconnaitre comme destinataire d'un message, lors de la transmission de données.

Un message de type $RREQ$ est de la forme :

$$< RREQ, ID_{REQ}, Flag, Trapdoor, Routing, PS_i >$$

| | |
|---|---|
| $RREQ$ | Message de type *Route Request* |
| $RREP$ | Message de type *Route Reply* |
| $ID_{REQ}$ | Identifiant unique de la requête |
| $Trapdoor$ | Trapdoor identitaire ou géographique |
| $Routing$ | Filtre de Bloom contenant les nœuds traversés par la requête |
| $PS_i$ | Pseudonyme temporaire du nœud $i$ |
| $L^k_{i,j}$ | $k^{ieme}$ identifiant de lien entre le nœud $i$ et le nœud $j$ |
| $S^k_{i,j}$ | $k^{ieme}$ clé partagée entre le nœud $i$ et le nœud $j$ |
| $Flag$ | Booléen indiquant si la trappe est de type identitaire ou géographique |
| $Challenge$ | Challenge généré par le nœud destinataire prouvant la bonne ouverture de la trappe |

FIGURE 9 – Notations

**Algorithm 1**: Gestion RREQ

**if** $ID_{REQ} \in RRT$ **then**
  **if** $(N_i||ID_{REQ}) \in Routing$ **then**
    `delete(`$msg$`);`
    `exit();`
  **end**
**else**
  `append(`$ID_{REQ}$, $PS_j$`);`
  `send(`$< RREQ, ID_{REQ}, Flag, Trapdoor, Routing \leftarrow (N_i||ID_{REQ}), PS_i >$`);`
**end**

Lorsqu'un nœud intermédiaire $i$ reçoit un message de type $RREQ$, il vérifie dans un premier temps si $ID_{REQ}$ est présent dans la table RRT. Si c'est le cas, cela signifie que la requête lui est déjà parvenue. Si son identifiant secret, noté $N_i$, concaténé avec l'identifiant de la requête $ID_{REQ}$ est présent dans le filtre de Bloom $Routing$, il supprime le message. Dans le cas contraire, ou si $ID_{REQ}$ n'est pas présent dans la table de routage, le nœud enregistre dans sa RRT l'identifiant de la requête $ID_{REQ}$ ainsi que le pseudonyme temporaire $PS_j$ présent dans le message de type $RREQ$. Il transmet ensuite le message $RREQ$ en remplaçant le pseudonyme temporaire présent dans le message par son pseudonyme temporaire $PS_i$ et en ajoutant son identifiant secret $N_i$ concaténé avec $ID_{REQ}$ au filtre de Bloom.

Le nœud $i$ teste ensuite s'il est le destinataire de la requête. Pour cela, il tente de déchiffrer avec sa clé privée $Priv_{dest}$ la trappe. S'il réussit, il est alors le destinataire du message et émet un message de type $RREP$.

Un message de type $RREP$ est de la forme :

$$< RREP, PS_i, L^k_{i,i-1}, (Flag, ID_{REQ}, Challenge)_{S^k_{i,i-1}} >$$

**Algorithm 2**: Gestion RREP

$auth \leftarrow$ `authenticate(`$PS_j$, $PS_i$`)`;
$(LinkID, Secret) \leftarrow$ `generate(`$auth$, $a$`)`;
**if** $LinkID = L^a_{i,j}$ **then**
    `decrypt(`$(Flag, ID_{REQ}, Challenge)_{S^a_{i,j}}$, $S^a_{i,j}$`)`;
**else**
    `delete(`$msg$`)`;
    `exit()`;
**end**
Node $\leftarrow$ RRT($ID_{REQ}$);
$auth \leftarrow$ `authenticate(`$PS_i$, $Node$`)`;
$(LinkID, Secret) \leftarrow$ `generate(`$auth$, $b$`)`;
`send(`$< RREP, PS_i, LinkID, (Flag, ID_{REQ}, Challenge)_{Secret} >$`)`;

Lorsqu'un nœud intermédiaire $i$ reçoit un message de type $RREP$, il utilise tout d'abord le pseudonyme $PS_j$ présent dans le message $RREP$ et son propre pseudonyme $PS_i$ pour générer $L^a_{i,j}$. Si l'identifiant du lien est différent de celui présent dans le message de type $RREP$, alors le nœud supprime le message. Sinon, il génère la clé $S^a_{i,j}$ et déchiffre $(Flag, ID_{REQ}, Challenge)_{S^a_{i,j}}$.

Le nœud utilise l'identifiant de la requête et la table RRT pour récupérer le pseudonyme du nœud $k$ situé en amont . Il génère ensuite $L^b_{i,k}$ et $S^b_{i,k}$. $i$ enregistre ensuite dans sa table FRT la paire d'identifiants de liens $(L^b_{i,k}, L^a_{i,j})$. Enfin, le nœud intermédiaire chiffre $(Flag, ID_{REQ}, Challenge)$ avec la clé $S^b_{i,k}$, remplace le pseudonyme et l'identifiant du lien et transmet le message de type $RREP$ ainsi modifié.

Lors de l'envoi de données, le nœud source utilise sa FRT pour déterminer à quels nœuds envoyer son message. Il obtient ainsi les identifiants du liens le reliant aux nœuds voulus. Il choisit ensuite aléatoirement dans cette liste le prochain nœud. Un nœud intermédiaire va procéder à la même opération, en regardant dans un premier temps si l'identifiant du lien utilisé dans le message n'est pas présent dans sa TLT. Si c'est le cas, le nœud est le nœud destinataire.

Les données peuvent être chiffrées avec la clé partagée lié à l'identifiant du lien. Cela permet aux données d'être chiffrées et déchiffrées à chaque saut.

### 5.3 Analyse du protocole

NoName reprend en partie le fonctionnement du protocole MASK. Ainsi, de part l'utilisation de pseudonymes sur les liens entre les nœuds, l'identité de la source et des nœuds intermédiaires est bien protégés. L'utilisation d'une trappe évite de diffuser l'identité du destinataire.

La fuite d'informations concernant la position de la source et de la destination dans

MASK était liée à l'utilisation de chemins totalement disjoints, ceux-ci ayant deux points de convergence, la source et le destination. L'utilisation du filtre de Bloom pour construire des chemins partiellement disjoints permet d'éviter cette fuite d'informations, les chemins partiellement disjoints n'ayant pas que deux points de convergence.

Enfin, comme pour MASK, l'utilisation du MIX-net associé aux pseudonymes sur les liens et au chiffrement de saut en saut permet d'assurer l'inassociabilité entre la source et un message, entre la destination et un message et enfin entre deux messages.

D'un point de vue performances, l'utilisation de chemins partiellement disjoints augmente le nombre de routes possibles et permet ainsi une meilleure résilience que MASK.

# 6 Conclusion et perspectives

Durant ce travail, nous nous sommes intéressé au respect de la vie privée dans les réseaux ad hoc. Nous avons tout d'abord défini les propriétés attendues dans le cadre spécifique des protocoles de routage respectueux de la vie privée en nous appuyant sur les travaux de Pfitzmann. Nous avons ensuite proposé une classification des différents attaquants en nous basant sur le fait qu'un attaquant peut être local ou global et être passif ou actif.

En nous basant sur cette étude préliminaire, nous avons ensuite analysé les différents protocoles de routage ad hoc prenant en compte le respect de la vie privée. Pour palier aux limitations des protocoles actuels, nous avons proposé NoName, un protocole de routage ad hoc topologique réactif basé sur MASK. NoName s'adresse aux limitations de MASK en permettant notamment ainsi de respecter l'anonymat du destinataire (par l'utilisation d'une trappe). Il assure aussi le respect de l'anonymat de la position de la source et de la destination par l'utilisation d'un filtre de Bloom permettant de construire plusieurs chemins partiellement disjoints. Notons de plus que le filtre de Bloom permet aussi d'améliorer la résilience du réseau.

Nous comptons maintenant profiter des derniers mois de stage pour réaliser une analyse poussée d'un certain nombre de propriétés de NoName : propriétés de respect de la vie privée mais aussi résilience, bande passant utilisée, performances, impact sur la batterie, sécurité... Nous envisageons aussi d'améliorer le protocole. Pour ce faire, nous souhaitons étudier tout d'abord la possibilité d'une trappe à ouverture rapide car un des problèmes potentiels de performances dans NoName est le temps d'ouverture de la trappe. Une piste de recherche consiste en la création d'une double trappe. La première trappe, plus rapide à calculer, sert à discriminer les nœuds du réseau en deux groupes : ceux pouvant potentiellement ouvrir la deuxième trappe et ceux ne pouvant en aucun cas ouvrir la deuxième trappe. Cette discrimination permettrait de limiter le nombre de nœuds testant l'ouverture de la deuxième trappe. Une implémentation possible de cette double trappe (ou trappe à ouverture rapide) consiste à utiliser une fonction (une fonction de hashage tronquée par

exemple) sur l'identité du destinataire concaténée avec l'identifiant de la requête.

Dans un second temps, nous étudierons la possibilité de réaliser du routage géographique en se basant sur le protocole NoName. Pour cela, nous nous pencherons sur la création d'une trappe géographique c'est à dire une trappe contenant une zone de destination et seuls les nœuds présents dans cette zone peuvent ouvrir la trappe. Le problème sous-jacent à la création d'une trappe géographique est le géochiffrement (*geo-encryption*). Le géochiffrement consiste à créer un chiffrement prenant en compte la position géographique, rendant ainsi impossible de déchiffrer le message en dehors d'une zone géographique déterminée. Ce problème est encore assez peu étudié. Les solutions proposées [16] se basent sur un GPS "sûr" (qui n'est pas capable de mentir sur sa position). Nous envisageons de proposer une solution faisant des hypothèses moins fortes.

## Références


[1] B.H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7) :422–426, 1970.

[2] P. Bose, H. Guo, E. Kranakis, A. Maheshwari, P. Morin, J. Morrison, M. Smid, and Y. Tang. On the false-positive rate of bloom filters. *Information Processing Letters*, 108(4) :210–213, 2008.

[3] A. Boukerche, K. El-Khatib, L. Xu, and L. Korba. A novel solution for achieving anonymity in wireless ad hoc networks. In *Proceedings of the 1st ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous networks*, pages 30–38. ACM, 2004.

[4] A. Broder and M. Mitzenmacher. Network applications of bloom filters : A survey. *Internet Mathematics*, 1(4) :485–509, 2004.

[5] Levente Buttyan and Jean-Pierre Hubaux. *Security and Cooperation in Wireless Networks*. Cambridge University Press, Cambridge, 2008.

[6] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr). United States, 2003. RFC Editor.

[7] K El Defrawy and G Tsudik. Alarm : Anonymous location-aided routing in suspicious manets. *2007 IEEE International Conference on Network Protocols*, 10(9) :304–313, 2007.

[8] K. El Defrawy and G. Tsudik. Prism : Privacy-friendly routing in suspicious manets (and vanets). In *Network Protocols, 2008. ICNP 2008. IEEE International Conference on*, pages 258–267. IEEE, 2008.

[9] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. Show me how you move and i will tell you who you are. In *Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '10, pages 34–41, New York, NY, USA, 2010. ACM.

[10] David B. Johnson, David A. Maltz, and Josh Broch. Dsr : the dynamic source routing protocol for multihop wireless ad hoc networks. In *Ad hoc networking*, pages 139–172. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[11] Jiejun Kong and Xiaoyan Hong. Anodr : anonymous on demand routing with untraceable routes for mobile ad-hoc networks. In *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, MobiHoc '03, pages 291–302, New York, NY, USA, 2003. ACM.

[12] S.J. Lee and M. Gerla. Aodv-br : Backup routing in ad hoc networks. In *Wireless Communications and Networking Conference, 2000. WCNC. 2000 IEEE*, volume 3, pages 1311–1316. Ieee, 2000.

[13] W. Lou, W. Liu, and Y. Zhang. Performance optimization using multipath routing in mobile ad hoc and wireless sensor networks. *Combinatorial optimization in communication networks*, pages 117–146, 2006.

[14] M.R. Pearlman, Z.J. Haas, P. Sholander, and S.S. Tabrizi. On the impact of alternate path routing for load balancing in mobile ad hoc networks. In *Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on*, pages 3–10. IEEE, 2000.

[15] Andreas Pfitzmann, Marit Hansen, Tu Dresden, and Uld Kiel. Anonymity, unlinkability, unobservability, pseudonymity, and identity management a consolidated proposal for terminology. Technical report, 2005.

[16] L. Scott and D.E. Denning. A location based encryption technique and some of its applications. In *ION National Technical Meeting*, volume 2003, pages 730–740, 2003.

[17] S. Seys and B. Preneel. The wandering nodes : Key management for low-power mobile ad hoc networks. In *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pages 916–922. IEEE, 2005.

[18] S. Seys and B. Preneel. Arm : Anonymous routing protocol for mobile ad hoc networks. *International Journal of Wireless and Mobile Computing*, 3(3) :145–155, 2009.

[19] Ronggong Song, Larry Korba, and George Yee. Anondsr : Efficient anonymous dynamic source routing for mobile ad-hoc networks. In *Proceedings of the 2005 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2005)*, 2005.

[20] D. Sy, R. Chen, and L. Bao. Odar : On-demand anonymous routing in ad hoc networks. In *Mobile Adhoc and Sensor Systems (MASS), 2006 IEEE International Conference on*, pages 267–276. IEEE, 2006.

[21] A. Whitaker and D. Wetherall. Forwarding without loops in icarus. In *Open Architectures and Network Programming Proceedings, 2002 IEEE*, pages 63–75. IEEE, 2002.

[22] Y. Zhang, W. Liu, W. Lou, and Y. Fang. Mask : Anonymous on-demand routing in mobile ad hoc networks. *Wireless Communications, IEEE Transactions on*, 5(9) :2376–2385, 2006.

[23] B. Zhu, Z. Wan, M.S. Kankanhalli, F. Bao, and R.H. Deng. Anonymous secure routing in mobile ad-hoc networks. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 102–108. IEEE, 2004.