

N° d'ordre: 3828

THÈSE

présentée

devant l'Université de Rennes 1

pour obtenir

le grade de : DOCTEUR DE L'UNIVERSITÉ DE RENNES 1
Mention INFORMATIQUE

par

Hassiba-Asmaa ADNANE

Équipe d'accueil : Sécurité des Systèmes d'Information et des Réseaux (SSIR)
École Doctorale : Matisse
Composante universitaire : SUPÉLEC

Titre de la thèse :

*La confiance dans le routage Ad hoc :
étude du protocole OLSR*

À soutenir le 3 Décembre 2008 devant la commission d'examen

M. :	Michel	HURFIN	Président
MM. :	Maryline	MAKNAVICIUS-LAURENT	Rapporteurs
	Jean-marc	SEIGNEUR	
MM. :	Christophe	BIDAN	Examineurs
	Rafael	DE SOUSA JUNIOR	
	Ludovic	MÉ	

*La méfiance est la mère de la sûreté*¹

¹Jean De Lafontaine (1621–1695).

Je dédie ce travail à mes parents, mon frère Bachir, mes soeurs Nour-el-imen et Hafsa, et mon mari Saïd. Sans oublier tous les membres de ma famille et mes amis qui m'ont soutenu au cours de ces 4 dernières années.

Remerciements

Je remercie Michel HURFIN, Professeur à l'université de Rennes 1, qui me fait l'honneur de présider ce jury.

Je remercie Maryline MAKNAVICIUS-LAURENT, professeur de l'institut national des télécommunications, et Jean-marc SEIGNEUR, Professeur à l'université de Genève en Suisse, d'avoir bien voulu accepter la charge de rapporteurs.

Je remercie Rafael DE SOUSA, professeur à l'université de Brasilia, d'avoir bien voulu juger ce travail.

Je remercie enfin Christophe BIDAN, enseignant chercheur à SUPELEC, et Ludovic MÉ, Professeur de l'université de Rennes 1, qui ont dirigé ma thèse.

Table des matières

Table des matières	1
1 Introduction	3
2 La sécurité des réseaux MANet (Mobile Ad-hoc Network)	6
2.1 Introduction aux réseaux ad-hoc (MANet)	6
2.2 Fonctionnalités des réseaux ad-hoc	7
2.2.1 Protocoles de routage dans les réseaux ad-hoc	7
2.2.2 Le protocole OLSR	8
2.2.3 Le protocole AODV	12
2.3 Sécurité des réseaux ad-hoc	14
2.3.1 Vulnérabilités des réseaux ad-hoc	14
2.3.2 Exigences de sécurité des réseaux ad-hoc	15
2.3.3 Attaques sur le routage	15
2.3.4 Travaux antérieurs	18
2.3.5 Sécurité de OLSR	20
2.4 Conclusion	23
3 Analyse de la confiance implicite dans le protocole OLSR	24
3.1 Notions de confiance	24
3.2 Langage et notations	25
3.2.1 Langage d'expression des clauses de la confiance	25
3.2.2 Notations	27
3.3 Analyse des aspects de confiance implicites du protocole OLSR	28
3.3.1 La découverte de voisinage	29
3.3.2 La sélection des MPR	30
3.3.3 Le calcul de la table de routage	32
3.3.4 Illustration : attaque par fabrication de message HELLO	35
3.4 Synthèse et conclusion	37
4 Intégration du raisonnement sur la confiance pour la sécurité de OLSR	39
4.1 Introduction	39
4.2 Intégration du raisonnement sur la confiance dans le protocole OLSR	40
4.2.1 Vérification de la cohérence des informations de bases : les liens	40

4.2.2	Vérification de la cohérence de la sélection des MPR	43
4.2.3	Vérification de la cohérence de la topologie du réseau	47
4.3	Illustration : attaque par fabrication de message HELLO	55
4.4	Conclusion	58
5	Prévention et contremesures	59
5.1	Validation du voisinage avec l'identité prouvable	60
5.1.1	Présentation de <i>M-SOLSR</i>	60
5.1.2	Présentation de <i>P-SOLSR</i>	62
5.1.3	Aperçu de l'identité prouvable	64
5.1.4	Preuve de voisinage	69
5.2	Contremesures	70
5.2.1	Sélection efficace des voisins et des MPR	71
5.2.2	Distribution de la preuve de méfiance	72
5.3	Conclusion	76
6	Validation par simulations du raisonnement basé sur la confiance	77
6.1	Environnement de Simulation	77
6.1.1	Implémentation des règles de confiance et des contremesures	78
6.1.2	Implémentation des attaques	79
6.2	Résultats de simulation	81
6.2.1	Etape des contremesures	85
6.2.2	Analyse du taux de faux positifs	87
6.3	Conclusion	90
7	Conclusions	91
A	Raisonnement sur la confiance avec la notation de l'identité prouvable	94
B	Présentation de GloMoSim	97
B.1	Installation	97
B.2	Configuration	98
	Glossaire	101
	Bibliographie	106
	Table des figures	107

Chapitre 1

Introduction

Les réseaux mobile ad-hoc (MANet) sont un nouveau paradigme des réseaux mobiles. Ils se construisent par l'interconnexion de différentes entités mobiles inconnues et ne reposent sur aucune infrastructure fixe ou un contrôle centralisé. La coopération entre ces entités permet de maintenir les services du réseau.

La principale fonctionnalité des réseaux ad-hoc est l'opération de *routing*. Elle contrôle et gère le trafic des messages dans le réseau. L'objectif principale d'un protocole de routage pour un réseau ad-hoc est l'établissement correct et efficace d'itinéraires entre une paire de nœuds de sorte que des messages puissent être acheminés. Le protocole de routage permet aux nœuds de se connecter directement les uns aux autres pour relayer les messages par des sauts multiples.

Un défi principal dans la conception de ces protocoles est leur sécurité. En effet, en l'absence d'une entité centrale ou d'une infrastructure fixe, les solutions de sécurité classiques ne sont pas adaptées aux réseaux ad-hoc. Dans ces réseaux les vulnérabilités sont nombreuses : usurpation d'identité, routeurs égoïstes ou malveillants, fabrication, modification ou suppression du trafic réseau, ...etc. D'autant que chaque nœud représente un point de vulnérabilité dans le réseau, car chacun a un rôle important dans le bon fonctionnement général du routage. En particulier, si aucun mécanisme n'est mise en place pour permettre à chaque nœud de déterminer le bon fonctionnement et de vérifier la cohérence des données de routage, le nœud accepte les informations de routage venant de tous les autres nœuds du réseau. Par conséquent, un attaquant peut envoyer des messages déclarant des informations incorrectes sur le réseau, afin d'y mener ensuite des actions malveillantes.

Plusieurs efforts de recherche ont été entrepris ces dernières années visant à sécuriser l'opération du routage dans les réseaux ad-hoc. Cependant, les solutions proposées se basent souvent sur une entité centralisée, sur la coopération entre les nœuds ou se limitent à des problèmes précis de sécurité (e.g. confidentialité et intégrité des échanges) et ne prennent pas en compte le comportement malveillant qui ne respecte pas le protocole de routage. Dans ce mémoire, nous défendons donc la thèse suivante :

- Les mécanismes de sécurité doivent tenir compte de la nature décentralisée et auto-configurable des réseaux ad-hoc.
- Les mécanismes de sécurité doivent prendre en considération les vulnérabilités du protocole de routage utilisé.
- Les mécanismes de sécurité doivent permettre à chaque nœud du réseau de raisonner sur le comportement des autres nœuds.

Comme Marsh [Mar94], nous pensons que la notion de confiance, quoique implicite, est toujours présente dans le fonctionnement des protocoles, en particulier, entre les entités qui participent aux opérations de routage. Il est ainsi primordiale que les entités la prennent en compte de façon explicite. De plus, la gestion explicite de la confiance permet aux entités de raisonner avec et à propos de la confiance, les rendant ainsi plus robustes pour la prise de décisions concernant les autres entités.

Dans le cadre de notre travail, nous nous sommes intéressés au concept de la gestion de confiance (*trust management*) pour sécuriser l'opération du routage avec le protocole OLSR (Optimized Link State Routing Protocol) [CJ03]. Trois contributions étayent cette thèse.

En premier lieu, nous proposons une analyse de la confiance implicite dans le protocole OLSR. Dans cette analyse, nous prouvons que le fonctionnement de OLSR génère des informations et présente des règles implicites de confiance entre les nœuds. Nous détaillons également le processus de construction et les conditions d'établissement des relations de confiance et nous montrons comment elles peuvent être exploitées pour mener des actions malveillantes.

Dans la deuxième contribution, nous proposons un raisonnement basé sur la confiance pour permettre de vérifier le comportement des nœuds. Le raisonnement est basé sur des vérifications de cohérence des informations reçues. En intégrant ce raisonnement dans chaque nœud et sans modifier le protocole OLSR, il est possible d'évaluer le comportement des autres nœuds, de détecter les comportements malveillants, et donc de pouvoir décider de faire confiance ou non.

Notre troisième contribution permet de résoudre le cas d'incohérence et contrer les comportements malveillants. Nous présentons deux mesures complémentaires : la prévention et les contremesures. La prévention permet de pallier certaines vulnérabilités du protocole, et les contremesures traitent les comportements anormaux afin d'isoler et de stopper les nœuds malveillants. Ces solutions ne nécessitent que peu de modifications sur le protocole OLSR et peuvent être étendues selon le type d'attaque et les besoins des utilisateurs.

Plan de la thèse

Dans le chapitre 2, nous définissons les réseaux mobiles ad-hoc (MANet) et la problématique du routage. Ensuite nous discutons de la sécurité du routage ad-hoc et plus particulièrement à travers l'exemple des protocoles OLSR et AODV.

Nous nous intéressons plus en détails au protocole OLSR dans le chapitre 3. Dans

ce chapitre, nous analysons le protocole sous l'optique de la confiance dans le but de présenter le processus et les conditions de construction de la confiance implicite entre les nœuds OLSR.

Afin de résoudre les problèmes de sécurité dans OLSR, nous présentons dans le chapitre 4 un raisonnement basé sur la confiance permettant de vérifier le comportement des nœuds et détecter les anomalies de comportement, et cela en se basant sur les observations individuelles. Ensuite, nous détaillons dans le chapitre 5 des techniques de prévention et de contremesures pour pallier à certaines vulnérabilités de OLSR et pour stopper les attaques.

Le chapitre 6 est dédié aux résultats de simulation du raisonnement basé sur la confiance et de certaines contremesures.

Enfin, le chapitre 7 présente une conclusion générale des travaux de la thèse et les perspectives de recherche.

Chapitre 2

La sécurité des réseaux MANet (Mobile Ad-hoc Network)

2.1 Introduction aux réseaux ad-hoc (MANet)

Un réseau ad-hoc ou MANet, défini par la RFC 2501 [CS99], est un réseau créé par une réunion de plusieurs entités mobiles ne disposant pas d'infrastructure pré-existante et d'autorité centralisée. Les réseaux MANet sont un nouveau paradigme de communication sans fil pour des entités mobiles : les entités à portée radio communiquent directement via les connexions sans fil, tandis que celles qui sont éloignées comptent sur d'autres entités pour relayer leurs messages. La mobilité des entités cause des changements fréquents de la topologie du réseau. Ce type de réseaux est utilisé dans divers domaines : secours, militaires, réseaux domestiques, téléphonie mobile,...etc. A l'origine, le terme MANet est le nom d'un groupe de travail de l'IETF, chargé de normaliser des protocoles des réseaux ad-hoc de mobiles. Depuis la naissance de ce groupe de travail, le nom propre MANet est utilisé comme nom commun pour désigner un réseau ad-hoc, spécialement dans les pays anglophones. Pour notre part, nous allons utiliser le terme réseau ad-hoc pour désigner les MANet (réseaux ad-hoc mobile).

Les principales caractéristiques des réseaux ad-hoc sont les suivantes :

Topologie dynamique : les entités du réseau ad-hoc sont mobiles et se déplacent d'une façon libre. Par conséquent, la topologie du réseau peut changer, à n'importe quel moment, d'une manière rapide et aléatoire.

Bande passante limitée : la communication sans fil se base sur l'utilisation d'un médium de communication partagé. Ce partage fait que la bande passante réservée à un hôte est modeste.

Puissance de calcul limitée : le paramètre d'énergie doit être pris en considération dans tout contrôle fait par le système, car une partie ou l'ensemble des nœuds repose sur des batteries ou un autre moyen limité et autonome pour puiser leur énergie.

Sécurité physique limitée : le paramètre de sécurité est très important pour les réseaux ad-hoc comparé au réseaux filaires. Puisque chaque nœud est potentielle-

ment nécessaire au fonctionnement du réseau, l'attaque physique sur un élément peut compromettre le fonctionnement global. Notons cependant que la mobilité augmente la robustesse du réseau en évitant les faiblesses des approches centralisées.

Absence d'infrastructure : les réseaux ad-hoc se distinguent des autres réseaux mobiles par l'absence d'infrastructure et de toute entité centralisée. Chaque entité mobile est responsable d'établir et de maintenir la connectivité du réseau d'une manière continue, et aussi de participer au bon fonctionnement du réseau.

2.2 Fonctionnalités des réseaux ad-hoc

La principale fonctionnalité du réseau ad-hoc est l'opération de *routing*. Le routage est une méthode d'acheminement des informations à la bonne destination à travers un réseau de connexion donné. Le problème de routage consiste à déterminer un acheminement optimal des paquets à travers le réseau au sens d'un certain critère de performance.

Compte tenu des caractéristiques des réseaux ad-hoc, des protocoles de routages spécifiques ont été créés pour s'adapter à la topologie dynamique de ces réseaux. Dans les protocoles de routage, chaque nœud peut jouer le rôle d'un routeur en transmettant les paquets reçus vers la destination. Par ailleurs, le routage dans les réseaux ad-hoc doit être auto-organisable, afin d'assurer l'acheminement des paquets à tous les nœuds en tenant compte de leur mobilité. En effet, puisque chaque nœud joue le rôle de routeur, le routage doit être auto-configurable en s'appuyant sur des routeurs mobiles, ces routeurs sont libres de se déplacer de façon aléatoire et de s'organiser eux-mêmes arbitrairement, car la topologie du réseau peut changer rapidement et de façon imprévisible.

2.2.1 Protocoles de routage dans les réseaux ad-hoc

Les protocoles de routage peuvent être classés selon différents critères. Par exemple, le moment d'initialisation de la découverte de routes nous permet de séparer les protocoles de routage en deux familles : proactifs et réactifs. Dans les protocoles réactifs la demande de routage est envoyée à la demande : si un nœud veut communiquer avec un autre, alors il diffuse une demande de route et attend une réponse du destinataire (exemple AODV [PBRD03], DSR [JHM03]). A l'inverse, les protocoles proactifs maintiennent à jour leurs informations de routage de façon à avoir en permanence une vue générale de la topologie du réseau (exemple OLSR [CJ03],FSR [GHG02]).

Chacun de ces types de protocoles a des avantages et des inconvénients/vulnérabilités. Il est nécessaire de comprendre le fonctionnement global pour pouvoir comprendre les attaques sur ces protocoles. Dans la suite de ce chapitre, nous allons présenter OLSR et AODV comme exemples de protocole de chaque famille et les attaques qui les menacent. Les deux protocoles ont fait chacun objet d'une RFC et semblent être en cours de normalisation.

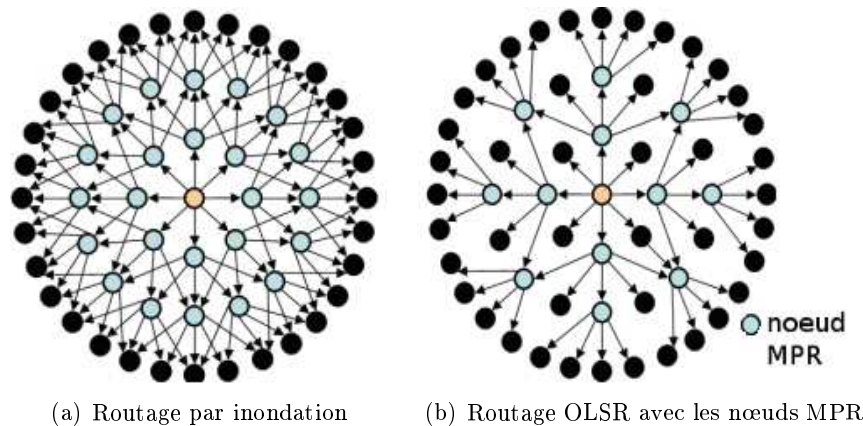


FIG. 2.1 – Routage optimisé avec OLSR

2.2.2 Le protocole OLSR

OLSR (Optimized Link State Routing protocol) est un protocole proactif à état de lien. Il est une variation du protocole LSR (Link State Routing) et il est conçu spécialement pour les réseaux ad-hoc. OLSR utilise un mécanisme d'inondation optimisée en propageant les informations de la topologie par l'intermédiaire de nœuds sélectionnés appelés relais multi-points (MPR : Multi Point Relay), pour permettre à tous les nœuds de communiquer entre eux et de s'échanger des informations sur leurs positions. Ainsi, le nombre de messages passant par les MPR se trouve réduit par rapport à une inondation classique, où chaque nœud retransmet chaque message quand il reçoit la première copie du message (figure 2.1 [WIK]). D'autre part, OLSR réduit le nombre de messages de contrôle diffusés dans le réseau, car l'information d'état de lien est produite seulement par les MPR. Chaque MPR, par ailleurs, ne doit rapporter que des liens entre lui-même et ses voisins qui l'ont sélectionné comme MPR (sélecteurs).

Dans OLSR, le nœud maintient sa *vision locale* du réseau dans une base d'informations appelée «*ensemble des associations d'interfaces*». Pour obtenir des informations du réseau et construire cette base d'informations, le protocole OLSR utilise deux types de messages de contrôle, HELLO et TC.

Les messages HELLO sont diffusés périodiquement par un nœud pour signaler sa présence et ses liens avec ses nœuds voisins (ces messages ne sont pas retransmis). L'échange de messages HELLO permet à un nœud la détection de différents types de voisins (symétriques, asymétriques ou MPR) et la construction des ensembles LinkSet (*LS* : inclut tous les voisins) et NeighborSet (*NS* : inclut seulement les voisins symétriques). La découverte de voisinage est présentée dans la figure 2.2.

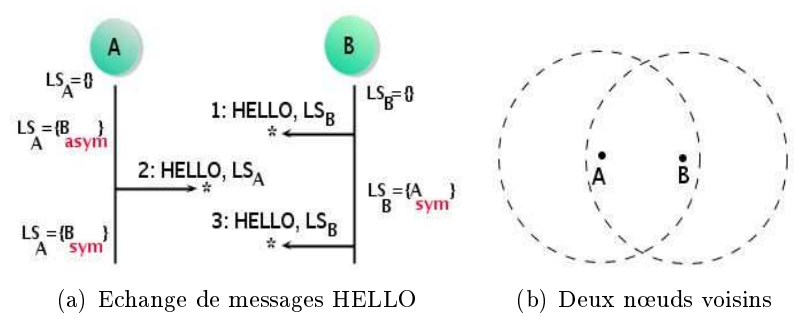


FIG. 2.2 – Découverte de voisinage par les nœuds OLSR

Au début des échanges, les nœuds ne possèdent aucune information sur leur voisinage, et leur ensemble *LinkSet* est vide. A la réception du premier message HELLO d'un nœud inconnu *B* (étape 1, figure 2.2), chaque nœud *A*, à portée radio de *B*, détecte un lien asymétrique avec ce voisin et l'annonce par la suite dans ses messages HELLO : $LS_A = \{(B, asym)\}$. De même, si *B* est à portée radio de *A*, il reçoit les message HELLO de ce dernier et il déduit que le nœud *A* est un voisin symétrique $LS_B = \{(A, sym)\}$ (étape 2, figure 2.2). Ensuite, le nœud *B* envoie de nouveaux messages HELLO annonçant son nouveau *LinkSet*. Lorsque *A* reçoit ces messages, il va déduire que *B* est un voisin symétrique et mettre à jour son *LinkSet* : $LS_A = \{(B, sym)\}$ (étape 3, figure 2.2).

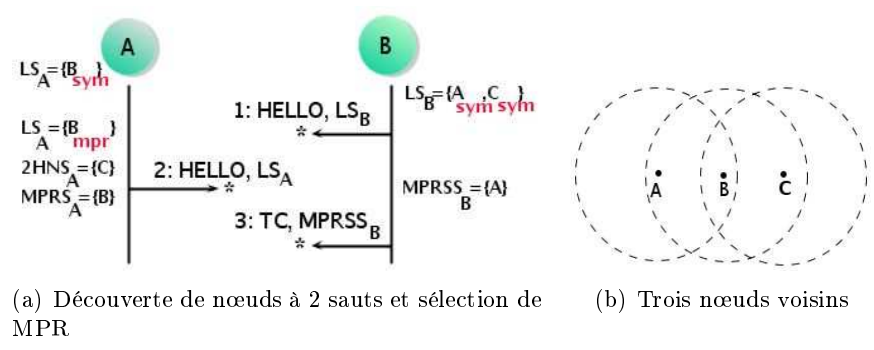


FIG. 2.3 – Découverte de voisins à 2 sauts

Dans les messages HELLO envoyés, les liens peuvent être déclarés comme asymétriques, symétriques ou MPR. Les messages HELLO reçus permettent à chaque nœud de mémoriser une base d'informations sur le voisinage à deux sauts (figure 2.3). Ces informations, qui composeront l'état mental interne de chaque nœud, sont représentées sous la forme d'ensembles, parmi lesquels les ensembles de liens (*Link Set* - *LS*), de nœuds voisins (*Neighbor Set* - *NS*), de nœuds voisins à 2 sauts (*2 Hop Neighbor Set* - *2HNS*), de nœuds sélectionnés comme MPR par le nœud en question (*MPR Set* - *MPRS*) et l'ensemble des voisins qui ont choisi le nœud comme MPR (*MPR Selector Set* - *MPRSS*).

Par exemple, dans la figure 2.3, lorsque A reçoit le message HELLO de B , il va détecter que ce dernier est un voisin symétrique de C , et ainsi déduire que C est un voisin à deux sauts. Puisque C n'est pas le voisin de A , ce dernier doit choisir son voisin B comme MPR pour pouvoir atteindre C , et doit déclarer B avec un statut MPR dans son message HELLO (étape 1, figure 2.3). A la réception des nouveaux message HELLO de A , le nœud B va déduire qu'il a été choisi comme MPR par A et doit donc l'ajouter dans son ensemble de sélecteurs MPR : $MPRSS_B = \{A\}$ (étape 2, figure 2.3), et générer un message TC annonçant être le MPR de A (étape 3, figure 2.3).

Les messages HELLO permettent ainsi à un nœud d'établir sa *vision locale* à 2 sauts. Par ailleurs, le message HELLO porte les ensembles LS, NS et MPRS d'un nœud, ce qui permet les opérations de découverte de liens, détection de voisins et signalisation de la sélection des MPR dans la *vision locale*.

Les informations accumulées dans ces ensembles sont actualisées et utilisées continuellement pour la sélection des MPR. Chaque nœud sélectionne ses MPR parmi ses voisins symétriques de telle façon à pouvoir accéder à tous ses voisins à deux sauts (exemple dans la figure 2.3). Chaque nœud mémorise aussi les adresses de ses voisins qui l'ont sélectionné comme MPR (ce qui constitue le $MPRSS$).

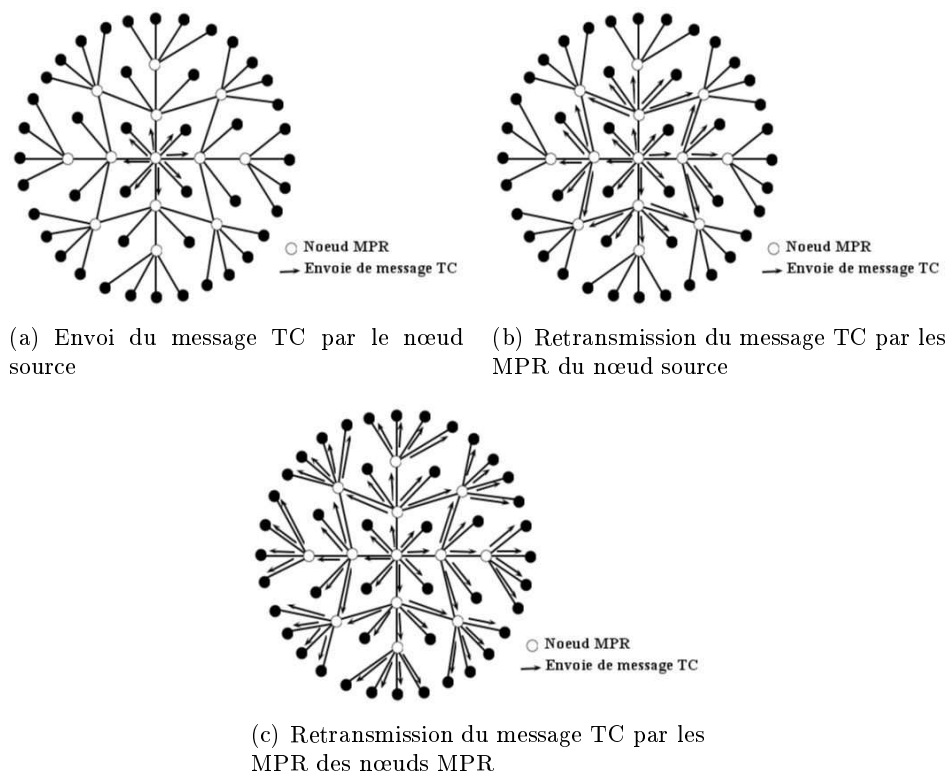


FIG. 2.4 – Génération et retransmission des messages TC

Le message TC porte les informations de topologie du réseau complet, c'est à dire la

vision globale. A partir de ces informations, chaque nœud maintient dans un ensemble nommé *TopologySet (TS)* les informations sur les points de destination dans le réseau. Un nœud qui a été sélectionné comme MPR émet périodiquement des messages TC annonçant la liste de ses voisins symétriques qui l'ont choisi comme MPR. Ce message TC n'est retransmis que par les nœuds qu'il a choisi lui-même comme MPR (exemple dans la figure 2.4). Les messages TC sont diffusés dans le réseau entier et permettent le calcul de la table de routage.

Considérant ces ensembles qui constituent l'état mental d'un nœud, et les prises de décisions concernant le passage de chacun des nœuds voisins d'un ensemble à un autre, la vision que le nœud a des autres nœuds peut être modélisée sous la forme du diagramme d'états et transitions de la figure 2.5.

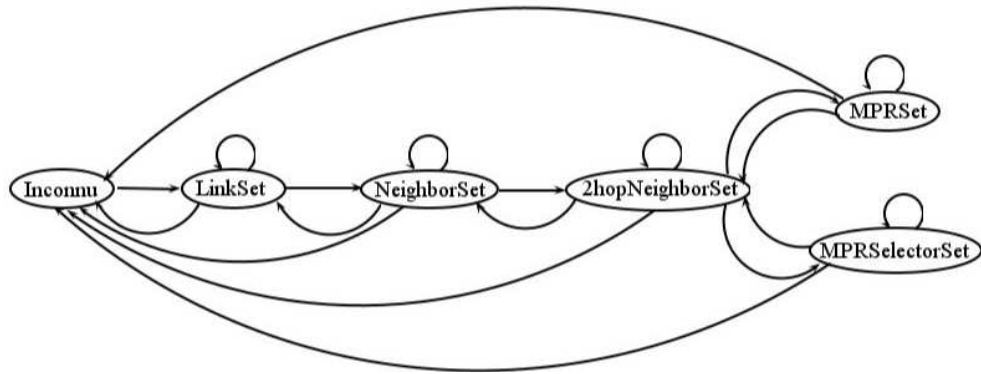


FIG. 2.5 – Vision possible du nœud OLSR sur chaque nœud de son voisinage à 2 sauts

En ce qui concerne les aspects de sécurité, la RFC 3626 ne spécifie pas de mesures de sécurité propres à OLSR. Cependant, les vulnérabilités du protocole y sont mises en évidence. Les problèmes soulevés font référence à la révélation de la topologie du réseau (car quiconque peut capturer les messages de contrôle), à la possibilité que des nœuds génèrent des messages de contrôle invalides (dans le but de donner une fausse vision de la topologie du réseau), à la possibilité d'interférences provenant de l'extérieur du domaine OLSR et à l'hypothèse d'unicité de l'adresse IP identifiant un nœud (une usurpation d'identité est facile à effectuer). Nous présentons plus de détails sur ce point dans la section 2.3.3.1.

Des solutions classiques de chiffrement et signature des messages, authentification de l'origine et horodatage des messages, ainsi que la restriction et le filtrage d'adresses, sont indiquées dans la RFC pour pallier ces problèmes de sécurité. En se basant sur ces techniques, deux versions sécurisées de OLSR ont été proposées par [ACJ⁺03] et [HTR⁺04]. D'autres travaux portant sur la sécurité de OLSR sont présentés dans la section 2.3.5.

2.2.3 Le protocole AODV

AODV (Ad-hoc On demand Distance Vector) est un protocole de routage pour les réseaux ad-hoc [PBRD03]. Il crée les routes au fur et à mesure des demandes et donc réduit le nombre de diffusions de messages. AODV est une amélioration du protocole DSDV (Destination-Sequenced Distance Vector [PB94]) qui est un protocole qui maintient la totalité des routes.

AODV envoie une requête de route sous la forme d'un message RREQ (Route Request) dans le but de créer un chemin vers une certaine destination, et il maintient les chemins d'une façon distribuée en gardant une table de routage au niveau de chaque nœud appartenant au chemin trouvé. L'envoi de messages entre un nœud source S et une destination D se déroule selon les étapes suivantes :

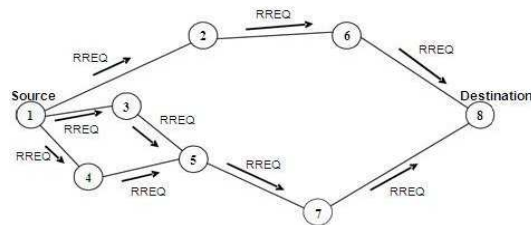
1. S diffuse un message de demande de route RREQ quand il a besoin de connaître une route vers la destination D et qu'une telle route n'est pas disponible (voir figure 2.6). Cela peut arriver si la destination D n'est pas connue au préalable, ou si le chemin existant vers la destination n'est plus valide ou défaillant. Ce message contient un identifiant (RREQ ID) unique pour tous les voisins. Cet identifiant est une incrémentation de la dernière valeur connue du numéro de séquence, associé au nœud destination. Cette valeur est recopiée de la table de routage.

Après la diffusion du message RREQ, la source attend le paquet réponse de route (RREP : Route REPLY). Si ce dernier n'est pas reçu durant une certaine période (appelée *RREP_WAIT_TIME*), la source peut rediffuser une nouvelle requête. A chaque nouvelle diffusion, le champ RREQ ID du paquet est incrémenté. Si la requête RREQ est rediffusée un certain nombre de fois (*RREQ_RETRIES*) sans la réception de réponse, un message d'erreur est délivré à l'application.

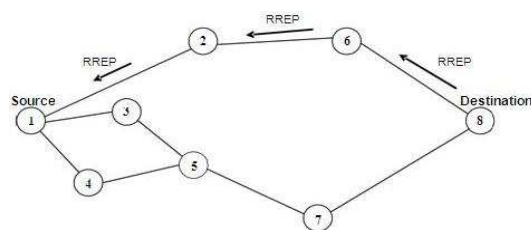
2. Quand un nœud intermédiaire reçoit le message RREQ, il met à jour le numéro de séquence (Sequence Number) dans sa table de routage pour pouvoir reconstruire ultérieurement le chemin inverse. Il sauvegarde aussi l'identité (i.e, l'adresse IP) du premier nœud qui lui a envoyé la requête. Si le nœud en question est la destination ou connaît un chemin qui mène à la destination et qui est considéré comme étant frais (*fresh route*), ce nœud envoie un message RREP (figure 2.6). Il est important de noter que les messages RREP sont émis en point à point (*unicast*) : un message RREP est envoyé au premier nœud ayant envoyé la requête RREQ correspondante. Chaque nœud sauvegarde l'identité du nœud qui a envoyé le RREP, car il sera considéré comme prochain saut vers la destination D .
3. Le nœud source ou le nœud intermédiaire qui reçoit le message RREP, met à jour le chemin qui mène à la destination dans sa table de routage.

AODV se base sur le principe des numéros de séquence pour pouvoir maintenir la consistance des informations de routage. Ce numéro indique la fraîcheur de l'information de routage et garantit l'absence de boucles de routages. Le numéro de séquence est incrémenté lorsque le nœud source initie une RREQ ou lorsque le nœud destination répond avec RREP.

Le chemin le plus court vers une destination est déterminé en utilisant le champs



(a) Propagation de la requête de demande de route RREQ



(b) Chemin pris par la requête de réponse (RREP)

FIG. 2.6 – Découverte de routes dans AODV

HopCount qui existe dans les paquets RREQ et RREP. Ce champ est incrémenté de 1 à chaque retransmission et représente le nombre de sauts entre la source et la destination dans la route proposée par le RREP correspondant.

Lorsqu'un lien est brisé, un message d'erreur RERR (Route Error) se propage vers la source dans le chemin inverse et tous les nœuds intermédiaires effacent l'entrée correspondante de leur table de routage. Afin de maintenir des routes consistantes, une transmission périodique d'un message « HELLO » est effectuée. Si trois messages « HELLO » ne sont pas reçus consécutivement à partir d'un nœud voisin, le lien en question est considéré défaillant. Les défaillances des liens sont, généralement, dues à la mobilité des nœuds dans le réseau ad-hoc.

En ce qui concerne les aspects de sécurité, la RFC de AODV ne spécifie pas de mesures de sécurité propres au protocole mais elle présente ses vulnérabilités. La construction des routes du protocole est la cible principale des attaques d'usurpation d'identité. Dans les réseaux où les nœuds sont inconnus, il est difficile de déterminer une attaque d'usurpation d'identité. Cependant, quand l'identité des nœuds est connue, les messages de commande de AODV peuvent être protégés en utilisant les techniques d'authentification comme la signature digitale. En particulier, il est important d'authentifier les messages RREP et RERR. Les messages RREP devraient être authentifiés pour éviter la création de faux itinéraires vers la destination recherchée. Dans le cas contraire, un attaquant pourrait prétendre être la destination, et faire un déni de service à la destination et/ou écouter le trafic envoyé à la destination. Les messages RERR

devraient également être authentifiés afin d'empêcher des noeuds malveillants de perturber des itinéraires valides. Nous présentons plus de détails sur les attaques contre AODV dans la section 2.3.3.2.

2.3 Sécurité des réseaux ad-hoc

2.3.1 Vulnérabilités des réseaux ad-hoc

L'utilisation de connexions sans fil rend le réseau plus vulnérable aux attaques, allant de l'écoute passive jusqu'aux attaques actives comme l'usurpation d'identité. L'écoute passive permet à un attaquant d'accéder à des informations secrètes (violation de la confidentialité). Les attaques actives permettent de supprimer des messages, d'injecter des messages erronés, d'usurper une identité,... etc violant ainsi la disponibilité, l'intégrité, l'authentification et la non-répudiation. Un noeud mobile dans un environnement hostile avec de faibles protections physiques a une grande probabilité d'être compromis. Par conséquent, nous devons prendre en considération les attaques malveillantes non seulement de l'extérieur, mais également au sein du réseau en considérant l'existence de noeuds compromis.

Après l'étude des vulnérabilités et menaces s'appliquant aux réseaux ad-hoc, l'article [VG03] dresse une liste des attaques les plus critiques. Ainsi, les attaques de type déni de services (denial of services DoS) semblent les plus faciles à réaliser. Elles deviennent très critiques dans certains contextes d'utilisation (secours, militaire). Voici quelques exemples de déni de services dans les réseaux ad-hoc :

- Brouillage du canal radio pour empêcher toute communication.
- Tentative de débordement des tables de routages des nœuds servant de relais.
- Egoïsme des nœuds : cela concerne les nœuds qui refusent de coopérer au bon fonctionnement du réseau, par exemple, pour préserver leur énergie. Ce problème est propre aux réseaux ad-hoc, car comme nous l'avons vu, le routage s'appuie sur la collaboration des nœuds. Un nœud refusant de jouer le jeu peut mettre en péril le bon fonctionnement du routage et donc du réseau ad-hoc.
- Tentative de gaspillage de l'énergie de nœuds ayant une autonomie de batterie faible ou cherchant à rester autonome (sans recharge) le plus longtemps possible. Ces nœuds se caractérisent par leur propension à passer en mode veille le plus souvent possible. L'attaque consiste à faire en sorte que le nœud soit obligé de rester en état d'activité et ainsi de lui faire consommer toute son énergie.
- Dispersion et suppression du trafic en jouant sur les mécanismes de routage.

Les attaques de type écoute et analyse du trafic constituent une menace pour la confidentialité des échanges. L'usurpation de l'identité d'un nœud en leurrant les mécanismes de contrôle d'accès permet de nombreuses attaques actives menaçant particulièrement l'opération du routage.

Enfin, il apparaît clairement que les attaques sur les mécanismes de routage sont particulièrement critiques. Nous accordons donc une large part à cette problématique dans la partie qui suit.

2.3.2 Exigences de sécurité des réseaux ad-hoc

A l'instar de la sécurité des réseaux traditionnels cablés, les exigences de sécurité des réseaux ad-hoc sont :

- La confidentialité des échanges : les informations ne sont accessibles que pour les entités autorisées. Sans ce mécanisme un nœud malveillant pourrait accéder aux informations secrètes contenues dans les messages transitants par lui.
- L'intégrité des échanges : un message transmis n'est jamais altéré. Sans ce mécanisme, un attaquant pourrait modifier le contenu des messages reçus avant de les retransmettre.
- L'authentification des entités : permet à chaque entité de garantir l'identité des autres entités avec qui elle communique. Sans ce mécanisme, un attaquant pourrait usurper l'identité d'une entité et ainsi changer les opérations de routage pour recevoir tous les messages à destination de l'identité usurpée.
- La disponibilité et la garantie de survie malgré une attaque de type déni de service (DoS).
- La non-répudiation des messages : l'origine d'un message ne peut pas nier avoir envoyé le message. Sans ce mécanisme il sera difficile de vérifier que l'émetteur et le destinataire sont bien les parties qui disent avoir respectivement envoyé ou reçu le message.

Cependant, il faut noter que, contrairement aux réseaux traditionnels qui peuvent s'appuyer sur une infrastructure centralisée fournissant des services réseaux sécurisés (e.g. le routage), il n'en est pas de même dans les réseaux ad-hoc. Par conséquent, les solutions de sécurité classiques ne sont pas adaptées aux caractéristiques des réseaux ad-hoc (mobilité, auto-organisation et absence d'une entité centrale ou d'une infrastructure fixe), et les nœuds eux-mêmes sont des points de vulnérabilités du réseau car un attaquant peut compromettre un élément laissé sans surveillance.

En conclusion, c'est pour cette raison que la plupart des attaques contre le routage exploitent les vulnérabilités de ces réseaux liées à la technologie sans fil sous-jacente et à l'absence d'infrastructure fixe.

2.3.3 Attaques sur le routage

Il existe plusieurs attaques sur l'opération de routage dans les réseaux ad-hoc, quelque soit le protocole de routage utilisé. Ces attaques exploitent les vulnérabilités et sont présentées dans la liste suivante :

- 1- Attaques du trou de vers (*Wormhole*) :** Le but de cette attaque est le détournement du trafic en utilisant un réseau privé. Pour lancer cette attaque, l'attaquant relie deux points éloignés dans le réseau avec un lien appelé lien de trou de ver. Ce lien peut être établi par plusieurs moyens, par exemple, en employant un câble d'Ethernet ou une transmission sans fil à longue portée. Une fois que le lien de trou de ver est établi, l'attaquant capture les transmissions sans fil sur une extrémité, les envoie par le lien de trou de ver et les rejoue à l'autre extrémité. Ainsi, l'attaquant établit un faux itinéraire qui raccourcit la distance en nombre de sauts

entre deux noeuds non-malveillants quelconques.

- 2- **Attaques du trou noir (*Blackhole*)** : Dans cette attaque, l'attaquant falsifie des informations (routes, liens et distances) afin d'être sélectionné dans une route active et pouvoir détourner ou absorber le trafic.
- 3- **Attaques par usurpation d'identité(s)** : L'attaquant falsifie les informations relatives à l'identité afin d'isoler un nœud (auquel il a volé l'identité) et donner une fausse vue de la topologie du réseau.
- 4- **Attaques par harcèlement ou déni de service** : Cette attaque s'effectue par émission régulière et inutile de paquets dans le but de surcharger le réseau et la consommation des ressources.

Ces attaques concernent tous les protocoles de routage. Il existe par ailleurs d'autres attaques qui exploitent précisément des vulnérabilités spécifiques à certains protocoles de routage. Nous présentons ces attaques pour OLSR et AODV.

2.3.3.1 Les attaques contre OLSR

En général, chaque nœud OLSR compte sur la collaboration des autres nœuds, et fait confiance de manière implicite à ses voisins pour participer à l'opération du routage. L'acceptation d'informations présentes dans les messages reçus, sans un mécanisme (par exemple, authentification) ou une procédure de validation (par exemple, vérification de la logique d'opération), est la vulnérabilité principale qui est exploitée par les attaques contre OLSR.

La référence [PMdSJ04] présente une classification de ces attaques (voir tableau 2.1). N'importe quel nœud peut modifier des messages du protocole pendant l'expédition, fabriquer de faux messages ou usurper une identité, et chacune de ces actions peut être à la base d'une attaque. Notons que le message HELLO est envoyé aux voisins à 1 saut seulement et n'est pas relayé. Il n'est donc pas concerné par les attaques de modifications, mais seulement par des attaques de fabrication. En revanche, le message TC est diffusé dans tout le réseau, et peut donc être utilisé pour des attaques du type modification (avant la réémission).

TAB. 2.1 – Les vulnérabilités du protocole OLSR

Attaque	Message OLSR	Informations faussées	Information sur l'origine du message corrompu
Fabrication	HELLO	liste des voisins	indifférent
Fabrication et usurpation d'identité	HELLO	état des liens	Adresse IP du nœud cible
Fabrication	TC	liste des voisins annoncés	indifférent
Modification et usurpation d'identité	TC	Numéro de séquence	Adresse IP de l'origine

Selon le but de l'attaquant, les différentes attaques contre OLSR peuvent être classées en deux groupes :

Attaques sur la sélection MPR : les attaques de cette catégorie ont pour but de perturber la sélection MPR. L'attaquant abuse des propriétés de l'algorithme de sélection des MPR (contenu et traitement des messages HELLO) afin d'être sélectionné comme MPR. Le plus grand risque de ces attaques est la sélection de l'attaquant comme unique et seul MPR du nœud cible, car l'attaquant pourra par la suite contrôler tous les flux de messages de ou vers la cible.

Attaques sur le calcul de la table de routage : Dans cette catégorie, le but de l'attaquant est de donner une fausse image de la topologie du réseau (messages TC) afin de perturber l'opération de routage.

Etre sélectionné comme MPR est la condition de base pour qu'un attaquant puisse réussir la plupart des attaques contre OLSR. En outre, si l'attaquant est MPR unique du nœud cible, cela présente la situation la plus critique pour la cible.

2.3.3.2 Attaques contre AODV

Dans AODV, chaque nœud compte sur la collaboration des autres nœuds, et leur fait confiance de manière implicite pour effectuer correctement l'opération du routage. AODV ne dispose d'aucun mécanisme de sécurité. Les nœuds malveillants peuvent effectuer plusieurs attaques seulement en détournant les règles du protocole. Un nœud malicieux peut effectuer les attaques suivantes contre AODV [NS03] :

Suppression du trafic de routage : Certains nœuds peuvent décider de ne pas participer au processus de routage et rejeter les paquets qui ne les concernent pas (un paquet qui ne lui est pas destiné ou qui n'est pas initié par lui). Ce comportement égoïste du nœud peut être dû à la limitation de la charge et de la capacité de calcul, ou à de la malveillance dans le but de perturber le réseau. Si une partie des nœuds est connectée au réseau par l'intermédiaire de ce nœud malveillant, ils deviennent inaccessibles et isolés.

Attaque par consommation de ressources : Cette attaque est effectuée par la génération et l'envoi fréquents de paquets de routage inutiles (paquets de type RREQ et RERR), dans le but d'inonder le réseau pour consommer toute la bande passante disponible avec un trafic non pertinent et ainsi faire consommer de l'énergie aux nœuds.

Attaque par modification de numéro de séquence : Dans cette attaque, l'attaquant répond à une requête RREQ en générant un paquet RREP avec un numéro de séquence grand pour favoriser sa requête. Ainsi, tous les nœuds vers la source vont insérer l'attaquant dans leur route active, et vont même participer à la propagation de fausses informations et ce en répondant à des futurs RREQ.

En général, AODV permet à des attaquants d'annoncer facilement de fausses informations d'itinéraire pour réorienter les routes et lancer des d'attaques. Dans chaque paquet de routage AODV, certains champs critiques tels que le nombre de sauts, numéro

de séquence de la source et la destination, et l'identification de RREQ, sont essentiels à l'exécution correcte du protocole. N'importe quel abus de ces champs peut perturber le fonctionnement correct de AODV.

2.3.4 Travaux antérieurs

Dans cette section, nous présentons les travaux génériques portant sur la sécurité des protocoles de routage des réseaux ad-hoc. Dans la section suivante (section 2.3.5, page 20), nous détaillons d'avantage les solutions spécifiques au protocole OLSR.

Beaucoup de travaux de recherche proposent des solutions pour sécuriser les protocoles de routage dans les réseaux ad-hoc. Certaines approches utilisent des mécanismes cryptographiques pour sécuriser des protocoles existants comme OLSR [RACM04], AODV [Zap05] et DSR [HPJ02], ou bien proposent de nouveaux protocoles sécurisés [DLRS02]. Cependant, ces techniques se sont seulement intéressées aux problèmes d'authentification, de confidentialité et d'intégrité, mais ne permettent pas de traiter le cas d'un nœud authentifié et malveillant.

La détection d'intrusion est un autre axe de recherche de la sécurité. Dans les réseaux ad-hoc, la plupart des travaux s'est concentrée sur la définition d'une architecture distribuée, où chaque nœud a un système de détection d'intrusion local (LIDS), et un système de détection global est mis en place grâce à la coopération entre les LIDS [ZLH00, PMdSJ04, PPMdS04]. Cependant, ces systèmes restent génériques, aucun travail n'a été porté précisément sur un protocole de routage en tenant compte de ses caractéristiques et ses vulnérabilités.

D'autres approches se sont intéressées aux problèmes de coopération entre les nœuds, car la coopération représente la base du fonctionnement du routage dans les réseaux ad-hoc. La plupart des solutions proposées se base sur les systèmes de réputation. L'un des premiers travaux propose le mécanisme du *watchdog* et du *pathrater* [MGLB00]. Le *watchdog* est utilisé localement par chaque nœud pour surveiller le voisin suivant dans le chemin de retransmission de messages, et si certains se conduisent mal et échouent dans la retransmission (par exemple, les nœuds égoïstes), la technique du *pathrater* est utilisée pour trouver un autre chemin vers la destination en évitant les nœuds qui se conduisent mal. Cependant, le fonctionnement du *watchdog* est basé sur des hypothèses qui ne sont pas toujours valides : carte réseau en mode promiscuité (*promiscuous mode*) qui est une configuration de la carte réseau lui permettant d'accepter le trafic à portée radio, même si celui ci ne lui est pas destiné. D'un autre côté, cette technique ne permet pas de partager les informations de réputation entre les nœuds en vue d'isoler et stopper les nœuds malveillants. Ces derniers peuvent ainsi continuer à utiliser les ressources du réseau même s'ils sont détectés.

La plupart des travaux qui ont suivi se repose sur le principe de la surveillance du comportement (*watchdog*). Par exemple, Michiardi & al. ont proposé un mécanisme appelé CORE (COLlaborative REputation mechanism), pour renforcer la coopération

entre les nœuds d'un réseau ad-hoc [MM02]. Ce mécanisme répond au problème de l'égoïsme des nœuds, il est générique et peut être associé à différentes applications distribuées. Il se base sur la surveillance distribuée où chaque entité encourage la collaboration d'autres entités en utilisant une métrique de coopération appelée *réputation* afin de noter le comportement des entités surveillées. Cependant, CORE n'empêche pas le comportement malveillant, mais s'assure que les entités se conduisant mal sont punies en leur refusant graduellement des services de communication. A noter que CORE donne la possibilité aux nœuds malveillants de réintégrer le réseau et de coopérer s'ils se comportent de nouveau correctement.

Dans l'article [BB02], les auteurs proposent un protocole appelé CONFIDANT, qui consiste en l'utilisation d'un système de réputation sur le protocole DSR. Les nœuds sont équipés d'un *watchdog* pour observer le comportement des voisins, et un avertissement est envoyé lors de la découverte d'un nœud égoïste sous la forme d'un message d'alarme. Après accumulation d'un certain nombre de messages d'alarmes, une décision est prise afin d'isoler le nœud égoïste.

Buttayan et Hubaux présentent un autre schéma appelé **Nuglets** [BH01], basé sur le système de marché, et une devise virtuelle appelée *nuglet*. Chaque nœud doit payer pour l'expédition de ses paquets et être payé pour expédier les paquets des autres. Les nœuds égoïstes vont finir leurs *nuglets* et ne pourront envoyer aucun paquet. L'inconvénient est que les *nuglets* sont gérés par une entité centralisée, et donc la méthode ne peut pas être aisément appliquée aux réseaux ad-hoc.

En résumé, les mécanismes de réputation se basent sur le partage d'information (les observations des autres nœuds), ce qui peut être une vulnérabilité dans le cas de fausses accusations. Par exemple, un nœud malveillant qui coopère dans les opérations du réseau ne sera jamais détecté, et peut même diffuser de fausses réputations sur d'autres nœuds légitimes.

De notre point de vue, même si les mécanismes de coopération et de réputation peuvent contribuer à la sécurisation des réseaux ad-hoc, ceux-ci doivent être renforcés par un système de gestion de la confiance entre les nœuds, pour détecter et isoler les nœuds malveillants (pas seulement les nœuds égoïstes) même s'ils coopèrent correctement.

La notion de confiance, ainsi que les modèles de confiance et de gestion de la confiance, ont fait l'objet de plusieurs recherches récentes. La confiance est reconnue comme un aspect important pour la prise de décision dans les applications distribuées auto-organisées [Mar94, YKB93, BCSC05]. Différents domaines se sont inspirés de ce concept : commerce électronique, réseaux P2P et également la sécurité du routage. Dans le domaine du routage, Xiaoqi *et al.* proposent un modèle de confiance pour sécuriser le protocole de routage AODV [XRM04]. Leur modèle permet aux nœuds de coopérer en vue d'obtenir un avis objectif sur la fiabilité de chaque nœud. Un autre modèle de confiance pour AODV a été proposé par [MVU06] : les auteurs proposent de séparer la confiance envers les nœuds et la confiance dans les routes choisies, leur but étant de mesurer la fiabilité pour atteindre n'importe quelle destination dans le réseau. Nous pouvons également citer les travaux de Yi *et al.* [YNK01] : après l'évaluation de la con-

fiance basée sur les observations locales, les auteurs proposent la prise en compte de la valeur de confiance dans les demandes de routes. De notre point de vue, ces travaux sont complémentaires, car chacun s'intéresse à l'évolution de la confiance dans les différentes étapes du fonctionnement du protocole.

Pour notre part, nous nous sommes intéressés particulièrement à la sécurité de OLSR, étant le protocole le plus utilisé dans la famille des protocoles proactifs.

2.3.5 Sécurité de OLSR

La spécification de OLSR [CJ03] ne définit aucune mesure spéciale de sécurité, mais reconnaît que, étant un protocole proactif, OLSR constitue une cible pour des attaques exploitant la diffusion périodique des informations topologiques. Plusieurs efforts ont été entrepris dans le but de trouver des solutions de sécurité pour OLSR. Dans cette section, nous présentons différents travaux et un aperçu de leurs principales caractéristiques ainsi que les hypothèses sous-jacentes.

Certains travaux s'appuient sur des mécanismes cryptographiques. Par exemple, dans [ACJ⁺03], les auteurs présentent une version sécurisée de OLSR (nous appelons *M-SOLSR* (*Message based secure OLSR*)). Leur mécanisme se base sur la signature et l'horodatage des messages de contrôle : la signature assure que le message vient d'un nœud de confiance, et l'horodatage permet de détecter la retransmission d'anciens messages. Les auteurs définissent deux types de nœuds : des nœuds de confiance (*trusted nodes*) qui ne peuvent pas être compromis, et des nœuds inconnus non-sûrs (*untrusted nodes*). Cependant, l'approche se limite à la détection de nœuds inconnus se comportant mal, et ne permet pas de détecter la situation où un nœud de confiance se comporte mal. D'un autre côté, il est difficile de définir le groupe des nœuds de confiance dans un réseau ad-hoc. Pour détecter les nœuds de confiance se conduisant mal, les auteurs proposent une technique avancée de signature pour sécuriser OLSR [RACM04]¹. L'idée de base est d'assurer une signature multiple par différents nœuds au trafic de contrôle (un message supplémentaire ADVSIG (*ADVanced SIGnature*)), afin de permettre à chaque nœud de prouver l'intégrité de son ensemble d'état des liens, et donc éviter que des nœuds compromis ne créent de fausses informations.

Les mêmes auteurs proposent une extension de leur travail [RACM05]. Le but est de détecter les attaques de retransmission et prendre des mesures contre les nœuds malveillants (par exemple, l'attaque du trou de vers (*wormhole*)[CLM⁺05]). Le comportement malveillant dans la retransmission de paquets est détecté en se basant sur le nombre de paquets envoyés et reçus par chaque nœud. Leur idée est basée sur la localisation géographique des nœuds, où la position géographique est obtenue à l'aide d'un GPS intégré dans chaque nœud, puis est insérée dans les messages de contrôle. Cependant, l'approche se limite à contrôler le nombre de paquets et ne permet pas d'assurer que les paquets ont été envoyés correctement.

¹Les travaux [ACJ⁺03, RACM04, CLM⁺05, RACM05] sont proposés par la même équipe.

D'autres solutions sont basées sur des approches cryptographiques. Par exemple Hafslunf & al. ont proposé une version sécurisée de OLSR (que nous appelons *P-SOLSR* (*Packet based secure OLSR*)) [HTR⁺04]. Leur idée est un peu similaire au travail proposé par [ACJ⁺03], la différence est qu'ils proposent l'authentification des messages en utilisant une signature numérique pour chaque paquet OLSR (et non pas chaque message comme proposé dans *M-SOLSR* [ACJ⁺03]).

Dans [KLMC08], les auteurs proposent une architecture AAA (Authentication, Authorization, Accounting) distribuée et hiérarchique basée sur OLSR appelée *WATCHMAN*. Leur but est de fournir une architecture légère et dynamique, où les composants de l'architecture AAA sont distribués sur les noeuds en utilisant différentes procédures d'élection. Dans cette architecture, chaque noeud est identifié avec sa paire de clés privée et publique, et dispose d'une autorisation de niveau de confiance (*ATL : Authorization Trust Level*) avec laquelle il peut accéder à différentes ressources du réseau. Les auteurs se basent sur deux types de serveurs : maître et esclave. Le serveur maître initialise le réseau et dispose des données d'authentification et d'autorisation. Il sélectionne un ou plusieurs serveurs esclaves parmi les noeuds du réseau et leur fournit les informations nécessaires (par exemple, liste des autres serveurs, leurs clés publiques, leurs autorisations et niveaux de confiance ...etc). La sélection se base sur différents critères : niveau de confiance, ressources, position dans le réseau. Ainsi, lorsqu'un serveur maître est indisponible, il est remplacé par le serveur esclave ayant les meilleurs critères. Cette architecture est adaptée aux domaines militaires, réseaux domestiques...etc.

Notons que dans la mesure où tous les noeuds sont authentifiés avant de participer aux activités du réseau, l'architecture *WATCHMAN* diminue les problèmes de sécurité du routage. Précisons également que les auteurs proposent d'utiliser le niveau de confiance (*ATL*) comme un critère de sélection des MPR afin de diminuer les problèmes de retransmission de trafic incorrecte et la génération de messages TC incorrectes.

Cependant, l'ensemble de ces travaux requiert une autorité centralisée, que ce soit pour la certification, pour la distribution des clés ou encore pour la détection des comportements malveillants. De notre point de vue, l'existence d'une telle autorité est incompatible avec la philosophie des réseaux ad-hoc.

Lebegue & al. se sont intéressés à la problématique de la sécurité de OLSR dans le domaine des réseaux tactiques militaires [LBP07]. Les auteurs ont proposé une extension de OLSR afin d'y intégrer le principe de groupe prédéfini. Ils ont ensuite défini les mécanismes d'authentification et d'échange de clés permettant aux noeuds d'un même groupe de s'authentifier mutuellement, et de générer le matériel cryptographique pour assurer la confidentialité et l'intégrité des échanges intra-groupe. Leur mécanisme permet de tenir compte des évolutions possibles des unités (pertes de noeuds, séparation, ou inversement, fusion de groupe).

Les mécanismes de système de réputations et de détection d'intrusions ont aussi été proposés pour sécuriser OLSR. Les travaux se sont intéressés particulièrement aux failles de sécurité face auxquelles les solutions cryptographiques sont inefficaces. Par exemple, Wang & al. [WLMG05] proposent une approche de détection d'intrusion basée sur la

vérification sémantique du protocole OLSR. Les propriétés sémantiques déduites par la définition du protocole sont utilisées par chaque nœud pour la détection d'incohérence par rapport au comportement correct décrit par la spécification de OLSR. Cependant, cette approche est très limitée, car elle ne concerne que les propriétés sémantiques de la *vision locale* et directe de chaque nœud et ne permet pas de partager les observations locales. D'autre part, les auteurs ne proposent pas de contremesures à prendre pour stopper et isoler les nœuds malveillants.

Cuppens *et al.*[CR07] ont précisément cherché à définir de telles contremesures contre les nœuds malveillants. En se basant sur la même technique de détection (propriétés sémantiques de OLSR), les auteurs analysent les attaques possibles contre OLSR, et présentent des techniques pour contrer l'attaque en proposant aux nœuds légitimes des routes qui n'incluent pas les nœuds malveillants. Toutefois, la détection reste à nouveau limitée à la *vision locale* et ne vérifie pas la cohérence de toutes ces observations (par exemple, ils ne proposent pas la corrélation entre messages TC, et la vérification de la table de routage). D'autre part, les auteurs n'ont pas vérifié si les contremesures et les nouvelles routes calculées restent cohérentes avec les propriétés sémantiques de OLSR, et ainsi garantir qu'elles ne vont pas être détectées comme un cas anormal.

En se basant sur le concept de la surveillance du comportement (*watchdog*), Vilela *et al.*[VB07] ont proposé un système de réputation pour OLSR. Leur mécanisme a pour but d'assurer la validité des messages de contrôle (message TC) générés dans le réseau. Afin d'atteindre cet objectif, à chaque réception d'un message TC, ce mécanisme génère un message de réponse appelé *feedback* qui contient le chemin parcouru par ce message, et l'envoi au nœud d'origine. À la réception du message *feedback*, les nœuds qui se conduisent mal sont pénalisés par la diminution de leur réputation ; inversement la réputation des nœuds se comportant bien est augmentée. Cependant, ce mécanisme se base sur la détection distribuée et ne tient pas compte du cas des mauvais témoignages. Effectivement, la réputation est partagée sans aucune validation. Un nœud malveillant qui se comporte correctement peut ainsi obtenir une réputation positive, pour ensuite fournir une mauvaise réputation sur d'autres nœuds légitimes se conduisant correctement afin de les pénaliser auprès des autres nœuds du réseau.

En résumé, les solutions proposées pour sécuriser OLSR ont différents inconvénients ou reposent sur des hypothèses parfois impossibles à établir dans un réseau ad-hoc. La solution serait de trouver un mécanisme qui assure l'intégrité des messages de contrôle (HELLO et TC), et qui permet de vérifier si le comportement des nœuds est correcte et respecte la spécification de OLSR afin de définir les nœuds de confiance. Le mécanisme de détection ne doit pas être centralisé, mais doit être intégré dans chaque nœud et doit se baser sur des données fiables (observations directes du nœud, et observations des nœuds de confiance). Ainsi, il est important d'analyser le comportement des nœuds et la construction des relations de confiance entre eux afin de pouvoir apporter des solutions sur la sécurité du routage OLSR.

2.4 Conclusion

Après l'étude des différents travaux sur la sécurité du routage dans les réseaux ad-hoc, et plus précisément sur le protocole OLSR, nous nous sommes intéressés au concept de la gestion de confiance (*trust management*). Ce concept s'adapte particulièrement à la nature mobile, ad-hoc, distribuée et auto-organisée des réseaux ad-hoc, où différentes entités communiquent sans aucune connaissance préalable sur l'identité et la nature des autres composants du réseau. De notre point de vue, le moyen le plus sûr de sécuriser chaque entité, est de lui permettre de raisonner sur le comportement des autres, en se basant sur ses observations locales et directes, afin de pouvoir décider si elle peut faire confiance ou non.

La confiance se révèle être un critère intéressant pour analyser les échanges entre des nœuds qui ne se connaissent pas. La première étape de notre travail a été d'analyser la confiance implicite dans le protocole OLSR et retrouver les points et les conditions des relations de confiance entre les différents nœuds qui échangent des informations de routage.

Chapitre 3

Analyse de la confiance implicite dans le protocole OLSR

Avant de voir comment le concept de la confiance peut être utilisé pour sécuriser OLSR, nous commençons par analyser la confiance implicite dans ce protocole. Cette analyse montre le processus de construction de la confiance entre les nœuds OLSR. Elle nous permet de voir les conditions d'établissement de la confiance pour ce protocole, ainsi que les attaques liées à la trahison/non-respect des relations de confiance.

3.1 Notions de confiance

Le concept de la confiance a été l'objet de plusieurs recherches dans différents domaines. Malgré cela, il n'y a pas de consensus dans la littérature sur la définition même de la confiance et sur ce que constitue la gestion de la confiance. Beaucoup de travaux de recherche proposent leurs propres définitions de la confiance, chacune concernant d'une manière très spécifique des domaines tels que l'authentification [YKB93], le commerce électronique [GS00, Vil05], le P2P [AD01] et bien d'autres domaines [BFL96].

Diego Gambetta présente la confiance comme «un niveau particulier de probabilité subjectif avec laquelle un agent peut évaluer un autre agent ou à un groupe d'agents effectuant une action particulière» [Gam80]. L'analyse des risques est considérée comme une partie importante de la gestion de confiance par le psychologue Morton Deutsch [Deu62]. Il définit la confiance comme un point de vue relatif qui est ciblée sur des aspects particuliers, et dépend des résultats et de la crédibilité de la personne concernée. Dans la même idée, certains travaux ont représenté la confiance comme un modèle qui doit être associé à la notion du risque, où l'évaluation de la confiance pour une entité tient compte du risque qu'elle représente. Par exemple, le projet SECURE [BDK⁺05, BCSC05, CGS⁺03] représente un modèle de collaboration qui se base sur l'évaluation de la confiance et la gestion du risque. Ce modèle permet de mesurer la confiance dans chaque entité à partir de son comportement et des recommandations qu'elle fournit. Ensuite, le module de la gestion du risque utilise la valeur de confiance pour estimer le bénéfice de la collaboration avec l'entité surveillée. Au final, la décision dépend du

bénéfice que peut apporter cette collaboration.

Quant à Steve Marsh [Mar94], il considère plus judicieux de s'intéresser au comportement de la confiance plutôt qu'à la confiance elle-même, enlevant ainsi la nécessité d'adhérer à une définition spécifique. La thèse de Marsh [Mar94] est l'un des travaux les plus cités dans la littérature de la gestion de la confiance. Dans ce travail, Marsh donne une attention particulière aux nombreuses facettes de la confiance, de la biologie à la sociologie, afin de développer un modèle de confiance entre les agents qui interagissent dans un mode distribué. Son modèle est complexe et très théorique, il reste générique et ne montre pas comment il peut être appliqué dans un domaine particulier.

Si ces travaux ont fait émerger une multitude de modèles formels de calcul et de gestion de la confiance, il en résulte en contrepartie une certaine confusion conceptuelle, mise en évidence par le fait que des concepts semblables apparaissent sous différents noms et réciproquement [GS00] [Vil05] [RK05]. Pour pallier à cette situation, nous considérons dans notre travail de recherche, comme Marsh[Mar94], qu'il est plus important de s'intéresser au comportement de la confiance plutôt qu'à la confiance elle-même. Plus précisément, chaque nœud OLSR, dans le réseau ad-hoc, se comportant selon la spécification de OLSR, telle qu'elle est décrite par la RFC3626 [CJ03], est considéré comme une entité se conduisant correctement et donc une entité de **confiance**. Pour exprimer le comportement de la confiance, nous utilisons le langage proposé par [YKB93], qui permet de formaliser et d'explicitier les aspects de confiance présents dans les protocoles de communications.

La première étape de notre travail est d'identifier et de formaliser les hypothèses de confiance utilisées implicitement dans le protocole OLSR. Un des buts de cette analyse est de proposer une extension au protocole OLSR de façon à le rendre plus résistant aux attaques, tout en évitant de poser des restrictions excessives concernant la capacité d'auto-organisation et la dynamique du réseau. Pour ce faire, nous partons de l'idée de classification de la confiance, qui consiste en une délimitation des circonstances où une relation de confiance est établie, et procédons à l'analyse des classes de confiance présentes dans ce protocole. Nous présentons en premier lieu le langage utilisé pour exprimer formellement les clauses de confiance, ainsi que la définition de la confiance sous-jacente à ce langage. Ensuite, nous exposons les caractéristiques générales et les problèmes de sécurité du protocole OLSR. Finalement, nous présentons les clauses de confiance implicites à OLSR et analysons les attaques contre ce protocole en fonction de ces clauses implicites.

3.2 Langage et notations

3.2.1 Langage d'expression des clauses de la confiance

Nous utilisons le langage proposé par [YKB93] pour l'expression des clauses relatives à la confiance dans un protocole. La notion de confiance sous-jacente à ce langage est exprimée par le fait que si une entité A a confiance en une entité B à propos d'un certain

aspect, cela signifie que A croit que B se comportera d'une certaine façon, réalisant (ou non) une certaine action dans des circonstances spécifiques.

La relation de confiance est prise en considération si la possibilité de réalisation d'une opération du protocole (l'action) est évaluée par l'entité A sur la base de ce qu'elle connaît de l'entité B et des circonstances de l'opération en question. En fonction de l'action qui est considérée et des circonstances d'exécution de cette action, il est nécessaire de distinguer différentes classes de confiance. Nous partons des classes de confiance définies par [YKB93] dans le contexte de l'analyse de protocoles d'authentification. Une entité peut avoir confiance en une autre pour une des actions suivantes :

- fournir l'identité d'une entité à une autre entité (classe *identification* : id) ;
- ne pas interférer dans les activités, fournir des données correctes, et ne pas modifier les données avant de les retransmettre (classe *non-interference* : ni) ;
- fournir des recommandations sur les caractéristiques de confiance d'autres entités (rec).

En outre, nous utilisons aussi les classes proposées par [GS00], spécifiquement :

- la confiance accordée par une entité pour l'accès à ses ressources ou services par d'autres entités (classe *access trust* : at) ;
- la confiance déléguée à une autre entité pour agir et prendre des décisions au nom d'une entité (classe *delegation trust* : dt).

D'autre part, dans le souci de précision de la formalisation des relations de confiance de OLSR, par rapport à ces classes, nous définissons des sous-classes correspondant à des actions réalisées effectivement par ce protocole, par exemple la confiance pour le routage.

Nous distinguons les relations de confiance directes et les relations de confiance dérivées, c-à-d établies à partir de recommandations d'autres entités. Étant données la présence de plusieurs types d'entités dans un environnement d'exécution du protocole et l'existence de rapports indirects entre ces entités, il est nécessaire de distinguer ces deux types de relations de confiance. Ainsi, les clauses relatives à la confiance sont exprimées avec la notation suivante :

- l'expression $A \text{ trusts}_{cc}(B)$ signifie que A a confiance en B pour réaliser l'action cc .
- $A \text{ trusts}_{cc}(ENS)$ signifie que A a confiance en toutes les entités appartenant à ENS pour réaliser l'action cc , ENS étant défini comme un ensemble d'entités pour lequel un certain prédicat est valable ;
- $A \text{ trusts}_{cc-C}(B)$ signifie que A a confiance en B pour réaliser l'action cc pour l'entité C , mais pas nécessairement pour d'autres entités ;
- $A \text{ trusts}_{rec_{cc}}(B) \text{ when.path}[ENS1] \text{ when.target}[ENS2]$ signifie que A fait confiance aux recommandations de B sur la capacité d'autres entités d'être dignes de confiance pour réaliser l'action cc . Les clauses *when* permettent de définir les con-

- traintes de ces recommandations, le *path* spécifiant un ensemble d'entités *ENS1* intermédiaires entre l'entité *B* et les entités cibles de la recommandation *ENS2* spécifiés dans le *target* ;
- $A \text{ trusts.rec}_{cc}^*(B)$ représente les recommandations dérivées, déduites à partir des précédentes recommandations ;
 - Dans le cas contraire, *A* n'a pas confiance en *B* est exprimé par $A\text{-trusts}(B)$. Dans notre travail, nous considérons que la *non-confiance*¹ est totale, et ne concerne pas une action particulière ; quand *B* trahit la confiance de *A* et n'accomplis pas correctement l'action qui a été demandée, alors *A* ne lui fait pas confiance pour accomplir d'autres actions. Nous appelons cette relation une méfiance *exacte*, car *A* détecte exactement le nœud malveillant *B* qui n'a pas accompli correctement l'action demandée ;
 - $A\text{-trusts}(ENS)$ signifie que *A* n'a pas confiance en les nœuds appartenant à l'ensemble *ENS*. Nous appelons cette relation une méfiance *partielle*, car *A* détecte un problème dans d'accomplissement d'une action demandée aux nœuds appartenant à l'ensemble *ENS*, mais il n'est pas capable de détecter quels nœuds précisément. Ainsi, il est important de noter que l'expression $A\text{-trusts}(\{B, C\})$ n'est pas équivalente à $A\text{-trusts}(B)$ et $A\text{-trusts}(C)$.

3.2.2 Notations

Dans OLSR, chaque nœud collecte des informations de configuration des liens (vision locale) et des informations de routage (vision globale) à partir des échanges de messages HELLO et TC, respectivement. Notre objectif est d'extraire des règles de confiance implicites et conjointement de suggérer que le protocole devrait aussi intégrer la notion de méfiance envers ses choix de MPR et de routes. Pour ce faire, nous partons des définitions suivantes :

- *MANET* est l'ensemble des nœuds du réseau MANET ;
- LS_x (Link Set) est l'ensemble des voisins du nœud *x*, il prend la forme suivante :

$$LS_x = MANET \times status, status \in \{asym, sym\}$$

Il inclut l'ensemble des voisins asymétriques (status=*asym*) et symétriques (status=*sym*) du nœud *x*.

- NS_x (Neighbor Set) est l'ensemble des voisins symétriques du nœud *x* ;
- $2HNS_x$ (2-Hop Neighbor Set) est l'ensemble des voisins à 2 sauts du nœud *x* ;
- $MPRS_x$ est l'ensemble des nœuds sélectionnés comme MPR par le nœud *x* ($MPRS_x \subseteq NS_x$) ;
- $MPRSS_x$ (MPR Selector Set) est l'ensemble des voisins symétriques qui ont sélectionné le nœud *x* comme MPR ;
- TS_x (Topology Set) est l'ensemble des informations sur la topologie du réseau vue par le nœud *x* ;
- RT_x (Routing Table) est la table de routage du nœud *x*, *RT* a le format suivant :

¹En anglais, il existe différents termes pour définir le contraire de la confiance : *distrust*, *mistrust*, et *untrust*. De notre part, nous avons choisi d'utiliser le terme *méfiance* pour exprimer l'état inverse de la confiance.

$(R_dest_addr, R_next_addr, R_dist)$, où R_dest_addr est l'adresse destination, R_next_addr est l'adresse du prochain voisin symétrique dans la route (qui doit être un MPR), R_dist est le nombre de sauts pour atteindre l'adresse destination² ;

- $Dist(x, y) : MANET^2 \rightarrow \mathbb{N}$ est la fonction qui fournit la distance en nombre de sauts entre deux nœuds du réseau x et y ;
- $route_{x \rightarrow y}$ est la séquence de nœuds qui composent la route (chemin le plus court) entre x et y , cette séquence a la forme du prédicat suivant :

$$route_{y_1 \rightarrow y_n} = y_1, \dots, y_n \text{ with } y_{i+1} \in MPR_{S_{y_i}}.$$

Le nœud collecte des informations pour construire une vision locale du réseau en échangeant des messages HELLO et TC. Ces échanges sont exprimés par les formules suivantes :

- $HELLO_x$: représente le message HELLO de x , il inclut l'ensemble des voisins de x . Dans ce message, chaque voisin est représenté avec l'expression suivante :

$$HELLO_x = MANET \times status, \text{ status} \in \{asym, sym, mpr\}$$

Par conséquent, ce message permet de déduire les ensembles LS_x, NS_x et $MPRS_x$:

$$\forall (y, asym) \in LS_x \Rightarrow (y, asym) \in HELLO_x$$

$$\forall y \in NS_x \Rightarrow (y, sym) \in HELLO_x$$

$$\forall y \in MPRS_x \Rightarrow (y, mpr) \in HELLO_x$$

- TC_x : représente le message TC du nœud x . Ce message inclut les voisins de x qui l'ont choisi comme MPR, et donc correspond à l'ensemble $MPRSS : TC_x = MPRSS_x$.
- $x \xleftarrow{HELLO_y} y$, $x \xleftarrow{TC_y} y$: respectivement, la réception par le nœud x des messages HELLO et TC du nœud y .
- $x \xrightarrow{TC_x} *$, $x \xrightarrow{DATA_x} *$: la diffusion par x d'un message TC ou d'un message de données devant être retransmis par ses MPR.
- $x \not\xleftarrow{TC_y} y$: l'absence d'un message TC attendu de la part du nœud y .
- $x \not\xrightarrow{DATA_x} y$: y est MPR de x , cette notation indique l'absence d'un message de données attendu, ce message étant généré par x et supposé être retransmis par y .
- $(TC_x)_y$: représente le message TC de x retransmis par y .

3.3 Analyse des aspects de confiance implicites du protocole OLSR

Étant données la description générale du protocole et la définition des ensembles maintenus par chaque nœud OLSR, il est possible d'utiliser le langage décrit dans la section précédente pour exprimer les relations de confiance dans ce protocole. En général, les nœuds sont considérés comme coopératifs et comme ayant confiance dans le fait d'obtenir la coopération des nœuds voisins. Cela correspond à la notion de confiance générale d'un agent définie par les travaux de Marsh [Mar94]. La table 3.1 montre quelques extraits de la RFC 3626 [CJ03] et donne leur traduction en utilisant le langage

²La table de routage inclut également l'adresse de l'interface utilisée pour atteindre R_next_addr

de spécification de la confiance. De la même façon, d'autres expressions similaires sont employées dans les sections suivantes pour l'analyse de la confiance implicite à OLSR et la description d'attaques contre OLSR.

TAB. 3.1 – Clauses générales de confiance d'un noeud OLSR

Expression de la RFC 3626	Notation formelle
Only nodes selected as MPR are responsible for forwarding control trafic (<i>fw</i>) (p. 4)	$N \text{ trusts}_{fw}(MPRS)$
The only requirement for OLSR to provide shortest path routes to all destinations is that the MPR nodes declare link-state information (<i>dlsi</i>) for their MPR selectors (p. 4)	$N \text{ trusts}_{dlsi}(MPRS)$
A node should always use the same address as its main address (p. 5)	$N_i \text{ trusts}_{id}(N_j), i \neq j$

Dans ce tableau, nous avons défini deux classes de confiance propres au protocole OLSR ; un nœud OLSR peut avoir confiance en ses *MPR* pour accomplir les actions suivantes :

- retransmettre ses paquets (classe *forwarding* : fw) ;
- déclarer correctement l'ensemble des voisins *LS* (classe *declare link-state information* : dlsi)

Dans cette section, tout en exprimant les règles de confiance implicites à OLSR, nous présentons des raisonnements sur la confiance qui peuvent être utilisés pour la découverte de voisinage, le calcul des MPR, la construction de la table de routage d'un nœud et également pour découvrir les vulnérabilités du protocole. Nous montrons ainsi que la confiance «explicitée» peut être un moyen de raisonnement pour le nœud à propos des informations qu'il reçoit [AdSJB07, dSJB07].

3.3.1 La découverte de voisinage

Initialement les nœuds sont généralement confiants [Mar94], puisqu'ils ne connaissent rien sur leur environnement et croient que toutes les informations reçues des autres sont correctes sans aucune vérification.

Au début, un nœud x ne connaît aucun voisin, donc il ne possède aucune vue sur le réseau. Il commence à construire sa vue à partir de la réception de messages HELLO provenant des voisins. La réception d'un message HELLO d'un nœud y permet de détecter des liens asymétriques, ce qui mène à une modification de l'état mental de x à propos de sa confiance en y , c-à-d x connaît y mais ne lui fait pas encore confiance, car il n'est pas sûr que y fonctionne conformément à la spécification de OLSR, en ce qui concerne la réception et l'envoi de messages HELLO :

$$\begin{aligned}
 x \xleftarrow{\text{HELLO}_y} y : & \quad (x, \text{asym}) \notin LS_y \implies x \neg \text{trusts}(y), LS_x \leftarrow LS_x \cup (y, \text{asym}) \\
 & \quad \text{et } (x, \text{sym}) \notin LS_y
 \end{aligned} \tag{3.1}$$

Cette expression signifie que x ne fait confiance à y ni pour être un voisin symétrique, ni pour être un MPR, bien que x reçoit des messages HELLO de la part de y . Cependant, étant un agent généralement confiant [Mar94], x diffuse des messages HELLO qui peuvent être reçus éventuellement par y , qui pourra les prendre en compte et ajouter x à son ensemble de voisins symétriques NS_y .

Si y se comporte de la façon attendue par le protocole, c-à-d s'il envoie des messages HELLO informant qu'il possède un lien avec x , alors on atteint une nouvelle situation de confiance :

$$\begin{aligned}
x \xleftarrow{HELLO_y} y, \quad (x, asym) \in LS_y &\Rightarrow x \text{ trusts}_{id \cup ni}(y), \\
\text{ou } (x, sym) \in LS_y & \quad LS_x \leftarrow LS_x \cup (y, sym) \\
& \quad \forall z \in NS_y, z \notin NS_x, z \neq x : \\
& \quad 2HNS_x \leftarrow 2HNS_x \cup \{z\} \quad (3.2)
\end{aligned}$$

Une relation de confiance vient d'être construite qui se concrétise par le fait que maintenant x considère y comme son voisin symétrique et les voisins symétriques de y comme voisins à 2 sauts. Cette relation de confiance est par ailleurs vue comme symétrique, puisque y devrait se comporter de la même façon que x :

$$\begin{aligned}
y \xleftarrow{HELLO_x} x, \quad (y, asym) \in LS_x &\Rightarrow y \text{ trusts}_{id \cup ni}(x), \\
\text{ou } (y, sym) \in LS_x & \quad LS_y \leftarrow LS_y \cup (x, sym) \\
& \quad \forall z \in NS_x - NS_y - \{y\} : \\
& \quad 2HNS_y \leftarrow 2HNS_y \cup \{z\}
\end{aligned}$$

Cette relation symétrique est la base pour des décisions qui seront prises par la suite par x à propos de sa vision locale (sélection des MPR), mais aussi, indirectement, concernant sa vision global pour le routage vers les nœuds distants (calcul de la table de routage) à travers l'échange des messages TC.

3.3.2 La sélection des MPR

La sélection des MPR est une étape importante dans le routage OLSR, où chaque nœud doit choisir parmi ses voisins ceux qui vont jouer le rôle de routeur pour lui. Dans OLSR, le seul critère de sélection des MPR pour un nœud x est le nombre de voisins à de deux sauts fournis par chaque voisin y , ce qui définit le degré de y , noté $D(y)$, et calculé par la formule suivante :

$$\forall y \in NS_x : V_y = NS_y - NS_x - \{x, y\} \text{ et } D(y) = \text{card}\{V_y\} \quad (3.3)$$

Au début, le nœud OLSR sélectionne comme MPR les voisins qui sont les seules offrant l'accessibilité/une route vers des voisins à deux sauts :

$$\begin{aligned}
MPRS_x = \quad MPRS_x \cup \{y \in NS_x : \exists Z \in 2HNS_x \text{ tel que } : \\
Z \in NS_y \text{ et } \forall V \in NS_x : Z \notin NS_V\} \quad (3.4)
\end{aligned}$$

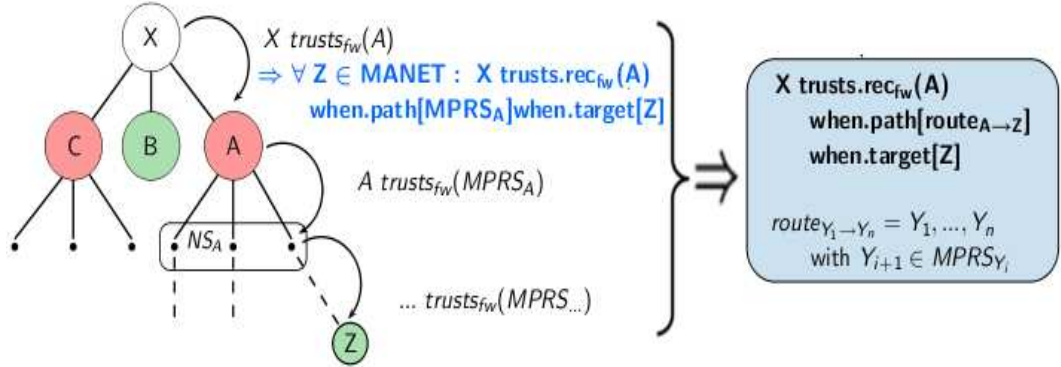


FIG. 3.1 – Chaîne de confiance entre les nœuds MPR

Ensuite, le nœud OLSR sélectionne le MPR ayant le degré le plus grand, jusqu'à ce que tous les voisins à deux sauts soient accessibles :

$$\begin{aligned} & \exists V \in 2HNS_x \text{ tel que } \forall y \in MPRS_x : V \notin NS_y \\ \implies & MPRS_x = MPRS_x \cup \{y \in NS_x : D(y) = \text{MAX}\{D(Z) | \forall Z \in NS_x\}\} \quad (3.5) \end{aligned}$$

En terme de confiance, quand x sélectionne des voisins comme MPR, cela signifie que x leur fait confiance pour être ses routeurs :

$$\forall y \in MPRS_x : x \text{ trusts}_{fw}(y) \quad (3.6)$$

Les voisins sélectionnés comme MPR choisissent également leur MPR et ainsi de suite. De cette manière, ils recommandent les chemins vers les nœuds distants aux voisins qui les ont sélectionnés comme MPR. En conséquence, chaque nœud fait confiance aux recommandations de ses MPR pour router vers n'importe quelle destination dans le réseau.

$$\forall Z \in MANET : x \text{ trusts}_{rec_{fw}}(y) \text{ when.path}[MPRS_y] \text{ when.target}[Z]$$

Comme les MPR du nœud y ($MPRS_y$) ont eux même confiance en d'autres MPR, la route de x vers la destination Z est formée par un enchaînement de MPR sous la forme du prédicat : $route_{y_1 \rightarrow y_n} = y_1, \dots, y_n$ avec $y_{i+1} \in MPRS_{y_i}$, ce qui permet l'extension de l'expression ci-dessus pour obtenir :

$$\forall Z \in MANET : x \text{ trusts}_{rec_{fw}}(y) \text{ when.path}[route_{y \rightarrow Z}] \text{ when.target}[Z] \quad (3.7)$$

Cette expression présente la règle générale de récursivité de la confiance pour le routage dans les réseaux opérant sous OLSR. Effectivement, le succès du routage ne compte pas seulement sur la sélection MPR locale, mais aussi sur la sélection MPR des

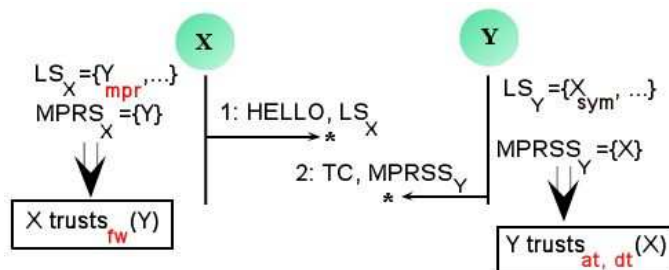


FIG. 3.2 – Diagramme de séquence de signalement des MPR

voisins. Par conséquent, chaque nœud doit aussi avoir confiance à la sélection des MPR de ses MPR, et ainsi de suite (figure 3.1).

Après la sélection des MPR, chaque nœud annonce son choix dans ses messages HELLO en modifiant le statut des liens. A la réception de ce message, les nœuds qui ont été choisis comme MPR vont mettre à jour leur ensemble de sélecteurs MPR ($MPRSS$). La séquence des échanges est représentée dans la figure 3.2.

Par ce calcul, un nœud y découvre les informations sur la confiance que d'autres nœuds mettent en lui. Le calcul du $MPRSS_y$ est exprimé par la formule suivante :

$$y \xrightarrow{HELLO_x} x : (y, mpr) \in HELLO_x \Rightarrow y \in MPRS_x, MPRSS_y \leftarrow MPRSS_y \cup \{x\} \quad (3.8)$$

Étant donné que y permettra aux nœuds de son $MPRSS$, c-à-d ceux qui l'ont choisi comme MPR, de l'utiliser comme un routeur, le calcul de $MPRSS_y$ implique les relations de confiance suivantes :

- y fait confiance en x pour utiliser ses ressources de routage sans causer de préjudices :

$$y \text{ trusts}_{at} (x)$$

- y fait confiance en x pour annoncer que y est un MPR :

$$y \text{ trusts}_{dt} (x)$$

La sélection des MPR et l'acceptation d'être choisi comme MPR représentent les relations de confiance de base pour le routage dans OLSR.

3.3.3 Le calcul de la table de routage

La table de routage représente le résultat du protocole OLSR, sur lequel s'appuie le routage. Le calcul de la table de routage est basé sur les informations fournies par les messages TC. La table de routage (RT) d'un nœud est mise à jour chaque fois qu'il y a un changement au niveau des ensembles : LS , NS , $2HNS$, TS , $MPRS$.

Rappelons que RT a le format suivant :

$$(R_dest_addr \mid R_next_addr \mid R_dist \mid R_iFace_addr)$$

où R_dest_addr est l'adresse destination, R_next_addr est l'adresse du prochain voisin symétrique dans la route (qui doit être un MPR), R_dist est le nombre de sauts pour atteindre l'adresse destination et R_iFace_addr est l'adresse de l'interface utilisée pour atteindre R_next_addr .

Chaque nœud x dispose d'une vue sur la topologie du réseau et cherche la route la plus courte pour atteindre n'importe quel autre nœud Z , cela via l'un de ses MPR (y). La table de routage est donc construite en utilisant l'algorithme du plus court chemin (Dijkstra) [Joh73]. Du point de vue de la confiance, ce calcul permettra à x d'avoir confiance en y pour le routage vers Z , s'il lui fournit le plus court chemin vers cette destination. En notant $T = (Z, y, n, I)$ un tuple de RT_x , la relation suivante est déduite :

$$\forall T \in RT_x \Rightarrow x \text{ trusts}_{fw-Z}(y) \quad (3.9)$$

La table de routage est calculée de telle sorte qu'il n'y ait qu'une seule route vers chaque destination :

$$\forall x, Z \in MANET \text{ et } Z \notin NS_x \Rightarrow \exists! T \in RT_x \text{ tel que : } T.R_Addr_Dest = Z \quad (3.10)$$

De plus, chaque route choisie est celle qui est la plus courte parmi les routes partant des nœuds MPR, ce qui définit un prédicat que nous appelons $MinDist(x, Z)$:

$$y \in MPRS_x : Dist(y, Z) = MIN\{dist(A, Z) / A \in MPRS_x\} \Rightarrow T.R_next_addr = y \quad (3.11)$$

Le risque inhérent au choix d'une seule route vers une quelconque destination est de choisir comme routeur un nœud corrompu ou malveillant. Dans la section suivante nous expliquerons comment cette vulnérabilité peut être exploitée par les attaquants, qui donnent des fausses informations sur la topologie du réseau afin de diriger tout le trafic du réseau vers eux en vue de perturber le fonctionnement du protocole.

Dans l'expression (3.11), même s'il existe plusieurs chemins vers Z , x ne va retenir qu'un seul chemin qui est le chemin le plus court fournis à partir d'un de ses MPR. Le calcul de la table de routage est un raisonnement du nœud basé sur la distance. Il en résulte l'ensemble des routes que le nœud considère comme les plus adéquates pour le routage. En fait, il s'agit dans ce calcul de faire le choix entre les MPR qui offrent des routes vers les destinations. Après le calcul des distances par le nœud lui-même, il fera plus confiance à celui qui offre la plus petite distance vers chaque destination (3.9).

Le choix par x d'un MPR y pour router vers un nœud Z implique que x fait confiance, non seulement en y pour router (3.6), mais aussi dans le choix des routes fait par y (3.7). En fait, il existe un enchaînement de relations de confiance indirecte entre x et tout relais acheminant le paquet vers Z , avec la particularité que seulement le dernier relais avant Z , étant son MPR, réalise des échanges directs avec Z (des messages HELLO). Cet enchaînement exprime la transitivité des recommandations des MPR existantes en OLSR, ce qui nous permet de déduire la relation suivantes :

Théorème 1. :

$$x \text{ trusts.rec}_{fw-z}^* (z) \text{ when.target}[z] \text{ when.path}[z]$$

Cette équation signifie que le point de départ (x) a confiance dans le nœud destination (z) pour pouvoir trouver une route pour l'atteindre, nous considérons que cette confiance concerne précisément l'opération de sélection des MPR. En effet, lorsqu'un nœud choisit correctement son ensemble de MPR, ces derniers vont diffuser un message TC annonçant son adresse à l'ensemble du réseau. Par conséquent, les nœuds distants pourront le détecter et construire un chemin pour l'atteindre. En résumé, la sélection des MPR est l'opération la plus importante et critique du protocole OLSR, permettant à la cible du routage de devenir elle-même le point de départ de la chaîne de confiance, chaque nœud doit ainsi bien choisir ses MPR afin que tout le monde puisse le localiser dans le réseau et communiquer avec lui correctement.

Démonstration. Soit $x, y_1, \dots, y_n \in \text{MANET}$ des nœuds du réseau. Si y_1 est le MPR de x et lui offre le plus court chemin vers la destination y_n , alors :

$$\exists T \in RT_x : T = (y_n, y_1, n, I_x) \text{ avec } n = \text{MinDist}(x, y_n) \Rightarrow y_1 \in \text{MPRS}_x$$

En appliquant les clauses 3.6 et 3.9, on obtient donc la règle suivante :

$$x \text{ trusts}_{fw-y_n}(y_1) \quad (3.12)$$

De plus, en appliquant la clause 3.7 on obtient :

$$x \text{ trusts.rec}_{fw}(y_1) \text{ when.path}[\text{route}_{y_1 \rightarrow y_n}] \text{ when.target}[y_n] \quad (3.13)$$

Par conséquent, les expressions 3.12 et 3.13 impliquent la clause suivante :

$$x \text{ trusts.rec}_{fw-y_n}(y_1) \text{ when.path}[\text{route}_{y_1 \rightarrow y_n}] \text{ when.target}[y_n] \quad (3.14)$$

Dans cette expression, x a confiance dans les recommandations de son MPR y_1 pour atteindre la destination y_n via la route $\text{route}_{y_1 \rightarrow y_n}$, cette route est constitué d'une suite de nœuds MPR et suit la propriété suivante :

$$\begin{aligned} & y_{i+1} \in \text{MPRS}_{y_i} \\ & \text{et} \\ & \forall y_i \in \text{Route}_{y_1 \rightarrow y_n} : \exists T_i \in RT_{y_i} : T_i = (y_n, y_{i+1}, n - i, I_{y_i}). \end{aligned}$$

A partir de l'expression (3.14), on obtient la nouvelle règle de récursivité suivante :

$$y_i \text{ trusts.rec}_{fw-y_n}(y_{i+1}) \text{ when.path}[\text{route}_{y_{i+1} \rightarrow y_n}] \text{ when.target}[y_n] \quad (3.15)$$

Dans [YKB93], les auteurs définissent la règle suivante :

$$\begin{aligned} & P_i \text{ trusts.rec}_x(Q_i) \text{ when.path}[Sp_j] \text{ when.target}[St_j] \\ \text{et } & Q_j \text{ trusts.rec}_x(Q_k) \text{ when.path}[Sp_k] \text{ when.target}[St_k] \\ & \text{et } Q_k \in Sp_j \\ & \downarrow \\ & P_i \text{ trusts.rec}_x^*(Q_k) \text{ when.path}[Sp_j \cap Sp_k] \text{ when.target}[St_j \cap St_k] \end{aligned}$$

De même, en appliquant cette règle sur 3.15 on obtient :

$$\begin{aligned}
 & x \text{ trusts.rec}_{fw-y_n} (y_1) \text{ when.path}[\text{route}_{y_1 \rightarrow y_n}] \text{ when.target}[y_n] \\
 \text{et } & y_1 \text{ trusts.rec}_{fw-y_n} (y_2) \text{ when.path}[\text{route}_{y_2 \rightarrow y_n}] \text{ when.target}[y_n] \\
 & \quad \downarrow \\
 & x \text{ trusts.rec}_{fw-y_n}^* (y_2) \text{ when.path}[\text{route}_{y_2 \rightarrow y_n}] \text{ when.target}[y_n]
 \end{aligned}$$

En utilisant la récursivité avec les règles (3.15) et (3.16), on déduit la clause suivante :

$$x \text{ trusts.rec}_{fw-y_n}^* (y_{i+1}) \text{ when.path}[\text{route}_{y_{i+1} \rightarrow y_n}], i \in [1, n - 1] \text{ when.target}[y_n]$$

Finalement, pour $i=n-1$, nous déduisons :

$$x \text{ trusts.rec}_{fw-y_n}^* (y_n) \text{ when.path}[y_n] \text{ when.target}[y_n].$$

□

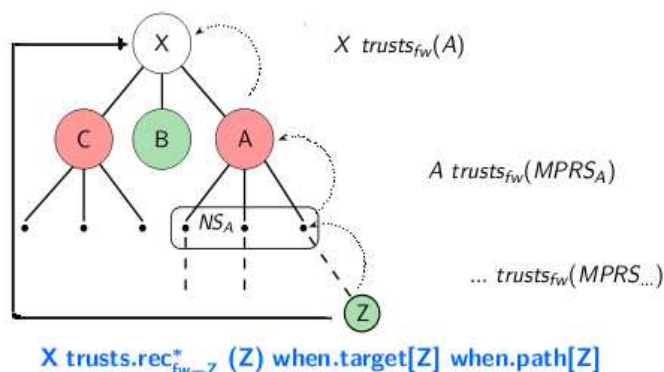


FIG. 3.3 – Propagation de la confiance entre les nœuds MPR

Cela suggère l'existence d'une propagation de la confiance accordée aux voisins MPR et une chaîne de recommandation entre les nœuds des routes calculées (figure 3.3). Certaines attaques exploitent la vulnérabilité due à l'absence de validation dans cette chaîne de confiance dérivée. Le nœud doit avoir un degré de méfiance concernant les informations qu'il utilise pour le calcul de la table de routage. Cette méfiance doit donc être associée à l'utilisation d'une procédure de validation des informations de routage qui circulent dans le réseau (messages TC).

3.3.4 Illustration : attaque par fabrication de message HELLO

Cette section représente l'étude d'un exemple d'attaque contre OLSR sous l'optique de la confiance. Dans cette attaque (figure 3.4), un nœud malveillant peut se faire élire comme MPR en créant un message HELLO annonçant tous les nœuds précédemment divulgués dans les messages HELLO qu'il a reçus, ainsi qu'un nœud fictif dont il est

le seul voisin symétrique. A la réception de ce message, les voisins de l'attaquant vont le choisir en tant que MPR unique (selon la règle 3.4). Ainsi tout le trafic sera expédié vers l'attaquant. Nous considérons que cette attaque est la plus fréquente contre OLSR, car elle représente la première étapes dans d'autres scénarios d'attaques contre ce protocole. En effet, il suffit d'être sélectionné comme MPR par la cible pour pouvoir contrôler/perturber les opérations de routage qui la concernent.

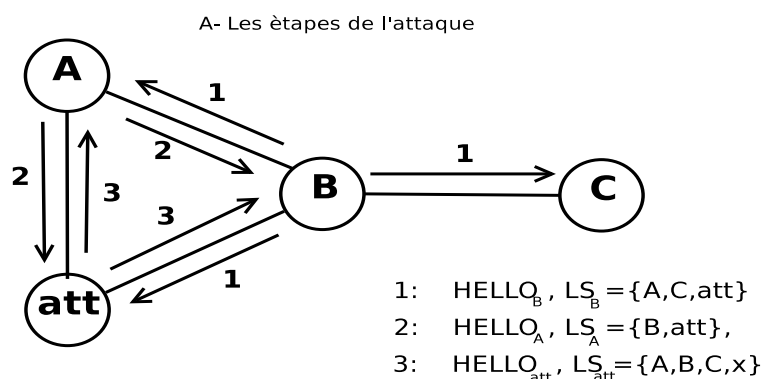


FIG. 3.4 – Fabrication de message HELLO

Avant l'attaque, A choisit B comme MPR pour transmettre les données à C . L'attaque se déroule selon les étapes suivantes :

1. $att \xleftarrow{HELLO} B$: l'attaquant identifie les voisins de B (A et C) comme voisins symétriques ;
2. $att \xleftarrow{HELLO} A$: l'attaquant identifie B comme voisin de A ;
3. l'attaquant fabrique un message HELLO annonçant les voisins symétriques de A et B :
 $\forall y \in NS_A \cup NS_B : LS_{att} \leftarrow LS_{att} \cup \{(y, sym)\} \cup \{x\} = \{A, B, C, x\}$
 où x est un nœud fictif déclaré comme ayant un lien symétrique avec l'attaquant ;
4. l'attaquant doit également générer un message TC annonçant les nœuds x et C . Effectivement, puisqu'il est le voisin unique du nœud fictif x , ce dernier doit le choisir comme son MPR pour accéder à A , B et C , et le nœud C doit également le choisir comme MPR pour accéder à x : $MPRSS_{att} = \{x, C\}$

En conséquence de cette attaque, selon la règle (3.4), A et B vont choisir att comme MPR, et vont donc lui faire confiance pour être leur routeur :

$$A \text{ trusts}_{fw}(att) \tag{3.16}$$

$$B \text{ trusts}_{fw}(att) \tag{3.17}$$

L'attaquant vient d'acquérir la confiance de A et B qui vont le choisir pour router vers n'importe quel nœud Z du réseau, sans avoir une preuve de l'existence d'un chemin entre l'attaquant et Z . Dans cet exemple, A va sélectionner l'attaquant pour router vers

C car il lui semble offrir le plus court chemin (3.11), ce qui aboutit à la situation exprimée par la règle (3.7) :

$$A \text{ trusts.rec}_{fw} (att) \text{ when.path}[route_{att \rightarrow C}] \text{ when.target}[C]$$

La non existence du chemin $route_{att \rightarrow C}$ annoncé par l'attaquant, montre que les nœuds (A et B) devraient se méfier des informations fournies dans le message HELLO. En effet, lorsque le nœud A envoie un message de données vers la destination C , ses voisins vont recevoir ce message, et seuls les voisins sélectionnés comme MPR vont le retransmettre. Dans cet exemple, le nœud B ne va pas retransmettre le paquet car il n'est pas le MPR de A , tandis que l'attaquant, qui est le MPR de A et devrait retransmettre tous ses paquets, ne va pas non plus retransmettre le message à destination de C dans le but de perturber et stopper les échanges entre A et C . Même si l'attaquant retransmettait ce message, il n'arriverait pas à C , car le lien symétrique entre att et C n'existe pas. Par conséquent, l'attaquant a trahit la confiance qui lui a été accordée de la part du nœud A (3.16), et ce dernier ne peut plus communiquer avec C : l'attaque est réussie.

D'un autre coté, le nœud C va sélectionner le nœud B comme MPR pour router vers A , construisant ainsi la relation de confiance 3.18 selon le théorème 1.

$$C \text{ trusts.rec}_{fw-A}^* (A) \text{ when.path}[A] \text{ when.target}[A] \quad (3.18)$$

Cette relation de confiance signifie que C a confiance dans le choix des MPR du nœud distant A afin de communiquer correctement. Néanmoins, cette relation est établie sans aucune vérification, et la confiance implicite posée dans le choix des MPR de A n'est pas méritée, car ce dernier a choisi ses MPR sans vérification, et ainsi, il a été la cible d'une attaque qui l'empêche de joindre le nœud C . Par conséquent, la réussite de l'attaque et la confiance que le nœud A a accordé à l'attaquant se sont répercutées également sur la relation de confiance entre A et C .

En résumé, la sélection d'un MPR représente un grand risque (surtout dans le cas d'un MPR unique), car la confiance accordée dans un MPR est la base du routage OLSR. Dans cet exemple, A prend un grand risque à choisir le nœud att comme MPR unique sans vérification : C et tous les voisins à plus d'un saut deviennent injoignables.

3.4 Synthèse et conclusion

Le protocole OLSR a pour objectif de calculer la table de routage, ce qui implique indirectement l'établissement de relations de confiance implicites entre les nœuds. En d'autres termes, OLSR génère des informations d'intérêt pour la confiance entre les nœuds, mais ceux-ci coopèrent d'abord et ensuite, sans preuves, déduisent implicitement des informations sur les autres nœuds en qui ils doivent avoir confiance. Le seul critère pour cette confiance est la distance entre les nœuds, aspect dont ils devraient se méfier. La méfiance est d'ailleurs un comportement qui serait plus approprié au début d'un rapport pouvant mener à une relation de coopération avec un inconnu. De plus, les

informations obtenues ne sont ni prises en compte pour les coopérations futures, ni exploitées pour améliorer le fonctionnement du protocole.

L'acceptation d'informations présentes dans les messages reçus, sans un mécanisme (par exemple, authentification) ou une procédure de validation (par exemple, vérification de la logique d'opération), est la vulnérabilité principale qui est exploitée par certaines attaques contre OLSR (présentées dans le chapitre précédent section 2.3.3.1). La confiance donnée à certains nœuds (et plus particulièrement les nœuds MPR) sans vérification représente un risque, car plus on fait confiance à un nœud, et plus sa trahison représente un risque important.

Deux résultats sont soulevés par cette analyse. En premier lieu, le fonctionnement de OLSR génère des informations et présente des règles implicites de confiance entre les nœuds qui ne sont pas prises en compte en tant que telles, mais qui peuvent être effectivement exploitées pour contribuer à la sécurité du protocole. En deuxième lieu, il s'avère que les nœuds créent des relations de confiance sans validation, ne mesurant pas les conséquences de ces relations et donc sans se méfier de leurs choix.

Chapitre 4

Intégration du raisonnement sur la confiance pour la sécurité de OLSR

4.1 Introduction

La notion de confiance, quoique implicite, est toujours présente dans le fonctionnement des protocoles. Il est déraisonnable que les entités ne la prennent pas en compte de façon explicite. En effet, gérer explicitement la confiance permet aux entités de raisonner avec et à propos de la confiance, les rendant ainsi plus robustes pour la prise de décisions concernant les autres entités [Mar94].

Le routage OLSR est le résultat d'un processus de coopération entre les nœuds pour découvrir les nœuds voisins, sélectionner les routeurs et annoncer les informations de topologie. Ce processus repose entièrement sur la confiance que chaque nœud a concernant les informations reçues des autres nœuds. Dans le chapitre précédent, l'analyse du protocole a soulevé le problème de la confiance implicite établi entre les nœuds sans aucune validation. Dans ce chapitre, nous proposons l'intégration dans le protocole OLSR d'un raisonnement basé sur la confiance afin de vérifier le comportement des nœuds. L'objectif est de construire un contrôle permettant à chaque nœud de détecter les anomalies de comportement des autres nœuds, et cela en se basant sur les observations locales.

En respectant la spécification de OLSR (RFC3626 [CJ03]), ce raisonnement pousse chaque entité à se méfier des nœuds inconnus, et leur permet d'observer le comportement des autres, et vérifier la cohérence de leurs informations avant d'établir une relation de confiance avec eux, et le cas échéant, de détecter les anomalies de comportement.

Dans la suite de notre travail, nous faisons l'hypothèse que l'identité des nœuds est garantie et ne peut être usurpée. Dans le chapitre 5, nous verrons comment cette hypothèse peut être assurée en nous appuyant sur la notion d'identité prouvable.

4.2 Intégration du raisonnement sur la confiance dans le protocole OLSR

L'analyse du protocole OLSR (chapitre 3, section 3.3) a montré que le nœud de destination du routage est lui-même le point de départ de la chaîne de confiance vers les autres nœuds. Ainsi, chaque nœud de destination doit bien choisir ses MPR afin que tous les autres nœuds puissent communiquer correctement avec lui. La sélection des MPR d'un nœud est ainsi une opération critique qui nécessite un mécanisme efficace de validation.

En outre, parce que certaines attaques contre le protocole OLSR exploitent la vulnérabilité résultant de l'absence de validation de la chaîne de confiance existante entre les MPR, les nœuds doivent avoir un degré de méfiance concernant les informations utilisées pour le calcul de la table de routage. Cette méfiance peut être associée à l'utilisation d'une procédure de validation des informations de routage qui sont distribuées dans le réseau.

Dans cette section, nous mettons en avant les propriétés et les relations de confiance au sein du protocole OLSR et cherchons à savoir comment un nœud peut détecter les anomalies de comportement par la méfiance envers les informations reçues du réseau. La détection d'anomalies comprend la vérification de cohérence des messages OLSR (messages TC et HELLO) et le raisonnement sur la confiance qui peut être effectué par chaque nœud du réseau. Bien que ce soit un processus continu, la détection doit progresser de la réception des messages de découverte de liens jusqu'à la construction de la table de routage, compte tenu de l'évolution de la confiance entre les nœuds au cours de ces opérations.

Nous présentons le raisonnement sur la confiance pour les trois étapes du protocole OLSR, à savoir : la validation des informations locales recueillies des voisins à 1 saut, la validation de la sélection de MPR et la validation de la table de routage.

4.2.1 Vérification de la cohérence des informations de bases : les liens

4.2.1.1 Validation de la vision locale

La notion de confiance au sein de OLSR est exprimée par le fait que, selon la spécification du protocole [CJ03], un nœud OLSR est convaincu qu'un voisin présente une bonne identification (*ID*) et s'engage à ne pas interférer (*NI*) dans l'exécution correcte du protocole. Par conséquent, comme l'a souligné [WLMG05], cela conduit à des propriétés intrinsèques du protocole en ce qui concerne le comportement attendu dans le traitement des messages et l'organisation du routage. Concrètement, un nœud x peut croire qu'un voisin y se comporte conformément à la spécification de OLSR si, en corrélant les messages en provenance de y , le nœud x est en mesure de vérifier les propriétés suivantes :

1. Les sélecteurs MPR d'un nœud y ($MPRSS_y$) annoncés dans le message TC_y sont des voisins symétriques ou MPR de y . Il doivent donc être déclarés comme tel dans le message HELLO du même nœud y . Lorsqu'un nœud x reçoit le message

HELLO de y , il peut déduire l'ensemble des voisins symétriques et MPR de y , et vérifier si les nœuds qui ont sélectionné y comme MPR (les nœuds inclus dans $MPRSS_y$ annoncés dans le message TC_y) ont été annoncés comme des voisins symétriques ou MPR par y :

$$x \xleftarrow{HELLO_y} y, x \xleftarrow{TC_y} y \Rightarrow MPRSS_y \subseteq NS_y$$

2. Un nœud y qui déclare x dans son message TC, et donc comme étant MPR de x , doit être perçu comme un voisin MPR par ce dernier. Lorsqu'un nœud reçoit le message $HELLO_x$, il va déduire l'ensemble $MPRS_x$, et ainsi vérifier si les nœuds qui déclarent x comme un sélecteur MPR dans leur message TC, ont bien été choisis comme MPR par x . Cette vérification est possible pour x lui-même, et pour les nœuds appartenant à la portée radio de x car ils sont les seuls à pouvoir recevoir $HELLO_x$ ($\forall z \in MANET : (x, asym \text{ ou } sym \text{ ou } mpr) \in LS_z$) :

$$x \in TC_y \Rightarrow y \in NS_x \text{ et } y \in MPRS_x$$

3. Lorsqu'un message TC ou DATA d'un nœud y est retransmis, il doit être identique à celui qui a été généré par y :

$$x \xleftarrow{TC_y} y, x \xleftarrow{(TC_y)_z} * \Rightarrow TC_y = (TC_y)_z$$

En règle générale, les messages TC ou DATA retransmis et ayant le même numéro de séquence, ne doivent pas être modifiés avant la retransmission :

$$\forall z, w \in MANET, x \xleftarrow{(TC_y)_z} *, x \xleftarrow{(TC_y)_w} * \Rightarrow (TC_y)_z = (TC_y)_w$$

Les possibilités pour un attaquant d'abuser de ces propriétés constituent les vulnérabilités principales du protocole OLSR. Cependant, en utilisant la notion de confiance précédente, nous sommes en mesure de reformuler ces propriétés pour exprimer des règles de méfiance qui peuvent être utilisées par un nœud pour son auto-protection contre les voisins ayant des comportements anormaux. Ceci conduit à un mécanisme de méfiance :

1. Si un nœud y annonce être le MPR d'autres nœuds (message TC), mais qui ne les a pas déclarés comme des voisins symétriques dans ses messages HELLO, alors son comportement ne correspond pas à la spécification de OLSR. Ses voisins (car eux seuls reçoivent les messages HELLO) doivent donc se méfier de lui :

$$x \xleftarrow{HELLO_y} y, x \xleftarrow{TC_y} y, MPRSS_y \not\subseteq NS_y \Rightarrow x \neg trusts(y) \quad (4.1)$$

2. Si x détecte qu'un nœud distant y , qui n'est pas son voisin, l'annonce dans son message TC, alors il doit se méfier de lui, car son comportement ne respecte pas la spécification de OLSR :

$$x \xleftarrow{TC_y} *, x \in MPRSS_y, y \notin NS_x \Rightarrow x \neg trusts(y) \quad (4.2)$$

3. Si x détecte qu'un nœud y (voisin ou non), l'annonce dans son message TC mais qu'il ne l'a pas choisi comme MPR, alors il doit se méfier de lui, car son comportement ne respecte pas la spécification de OLSR :

$$x \xleftarrow{TC_y} *, x \in TC_y, y \notin MPR S_x \Rightarrow x \neg trusts(y) \quad (4.3)$$

4. Si x reçoit le même message TC (même numéro de séquence et adresse d'origine) de deux nœuds différents (ou de l'origine aussi), et détecte que les deux messages sont différents, alors il doit se méfier des deux expéditeurs :

$$\forall z, w \in MANET, x \xleftarrow{(TC_y)_z} *, x \xleftarrow{(TC_y)_w} *, (TC_y)_z \neq (TC_y)_w \Rightarrow x \neg trusts(\{z, w\}) \quad (4.4)$$

Effectivement, même si l'un des expéditeurs est l'origine du message TC_y , il se peut qu'il ait un complice qui a envoyé un message TC aux autres nœuds et qui soit différent de celui que y a envoyé à x .

Ainsi, en utilisant ce raisonnement, chaque nœud est capable de valider sa vision de base sur l'état et les types des liens qu'il a avec ses voisins.

4.2.1.2 Validation de la vision des voisins

Les propriétés précédentes peuvent être étendues aussi pour permettre aux nœuds de vérifier la vision locale de leur voisins à propos du réseau :

1. Si x détecte qu'un nœud distant y annonce l'un de ses voisins dans son message TC_y , mais que ce dernier ne déclare pas y comme voisin symétrique, alors il doit se méfier de lui, car son comportement ne respecte pas la spécification de OLSR :

$$x \xleftarrow{HELLO_z} z, x \xleftarrow{TC_y} *, \exists z \in NS_x \cap MPR S_y, y \notin NS_z \Rightarrow x \neg trusts(y) \quad (4.5)$$

2. Si x détecte qu'un nœud y (voisin ou non), annonce l'un de ses voisins dans son message TC mais que y n'a pas été choisi comme MPR par ce voisin, alors il doit se méfier de lui, car son comportement ne respecte pas la spécification de OLSR :

$$x \xleftarrow{HELLO_z} z, x \xleftarrow{TC_y} *, z \in NS_x \cap MPR S_y, y \notin MPR S_z \Rightarrow x \neg trusts(y) \quad (4.6)$$

De même, la découverte de voisins (les liens) doit être cohérente entre tous les nœuds. L'un des problèmes qui peut se révéler dans cette opération est la réception de messages HELLO contradictoires provenant de différents voisins. Cela peut être une situation temporaire si un groupe de nœuds est au début d'un processus de découverte. Mais, si la situation se pose après un processus d'initialisation, le nœud qui continue à recevoir des messages contradictoires doit se méfier des nœuds qui génèrent ces messages. La découverte du voisinage peut être validée par la vérification des liens annoncés dans les messages HELLO reçus, compte tenu du fait que si un lien symétrique entre deux nœuds est annoncé par un de ces nœuds, il doit également être annoncé comme tel par l'autre :

$$x \xleftarrow{HELLO_y} y, x \xleftarrow{HELLO_z} z, z \in NS_y \Rightarrow y \in NS_z \quad (4.7)$$

En termes de confiance, cela signifie qu'un nœud ne peut pas être certain de l'existence du lien entre ses voisins s'il reçoit des informations contradictoires concernant ce lien, ce qui conduit à l'expression de méfiance suivante :

$$x \xleftarrow{HELLO_y} y, x \xleftarrow{HELLO_z} z, z \in NS_y, y \notin NS_z \Rightarrow x\text{-trusts}(y, z) \quad (4.8)$$

Ainsi, il est possible de mettre en place des contrôles élémentaires pour la détection de voisins ayant des comportements anormaux, avec des vérifications susceptibles d'accroître la robustesse du protocole et de permettre la détection de possibles attaques directes. A ce stade, ces vérifications ne suffisent pas pour contrer d'autres attaques et ne permettent pas une coopération globale pour la détection de comportements anormaux.

Il convient de souligner que les raisonnements sur la méfiance ne permettent pas tout le temps l'identification précise des nœuds ayant des comportements anormaux, mais permettent de détecter une anomalie de comportement liée à un groupe de nœuds qui comprend l'attaquant, comme indiqué dans les expressions 4.4 et 4.8.

4.2.2 Vérification de la cohérence de la sélection des MPR

La sélection de MPR est la base du protocole OLSR. C'est une opération critique car elle permet à chaque nœud de choisir ses points d'accès à l'ensemble du réseau. Elle permet également à chaque nœud d'être connu par les nœuds distants. Dans la spécification du protocole OLSR [CJ03], le comportement d'un MPR n'est pas contrôlé après sa sélection. Cette vulnérabilité est exploitée par de nombreuses attaques où l'attaquant cherche à être sélectionné comme MPR par un nœud cible [HP04], afin que le MPR attaquant puisse contrôler le flux d'entrée-sortie de la cible.

La vulnérabilité d'une sélection MPR fondée sur le degré d'accessibilité (des voisins à 2 sauts) vient du fait que tout attaquant peut donner de fausses informations et que ces informations ne peuvent pas être vérifiées. Il est ainsi possible à un attaquant de créer par exemple des voisins fictifs ou d'annoncer des nœuds distants comme voisins symétriques.

Pour renforcer la sélection de MPR, nous proposons donc que chaque nœud vérifie les deux points suivants :

1. les nœuds sélectionnés comme MPR doivent se comporter correctement en ce qui concerne les opérations de génération de messages TC, et la retransmission des messages TC et des paquets de données des nœuds qui l'ont sélectionné comme MPR ;
2. les choix locaux de MPR par un nœud doivent être conformes avec les informations globales de la topologie et cohérents avec les autres voisins.

Cela implique que le nœud doit contrôler les informations locales observées et enregistrées, et en particulier corrélérer les messages HELLO et TC comme décrit dans les paragraphes suivants.

4.2.2.1 Validation de la sélection MPR locale

Nous proposons que chaque nœud supervise le comportement de ses MPR. Dans le chapitre précédent, nous avons présenté la confiance entre un nœud et son MPR par l'expression suivante (chapitre 3, page 31, expression 3.6) :

$$\forall y \in MPRS_x : x \text{ trusts}_{fw}(y)$$

Cela signifie que x fait confiance aux nœuds dans son ensemble MPRS pour relayer des messages (FW). Selon la spécification du protocole OLSR, le bon comportement d'un MPR, en ce qui concerne le routage, est défini par deux opérations :

1. **Génération du message TC.** Chaque nœud sélectionné comme MPR doit produire périodiquement un message TC annonçant tous les nœuds qui l'ont choisis comme MPR (et ce message est entendu par ces nœuds) :

$$y \in MPRS_x \Rightarrow x \xleftarrow{TC_y} y, x \in TC_y \quad (4.9)$$

2. **Retransmission des paquets de données et des messages TC.** Chaque MPR doit rediffuser les paquets de données et les messages TC envoyés par les nœuds qui l'ont choisis comme MPR :

$$y \in MPRS_x \Rightarrow MPRSS_x \neq \emptyset : x \xleftarrow{(TC_x)_y} y \text{ et } x \xleftarrow{(DATA_x)_y} y \quad (4.10)$$

Si un nœud est en mesure de valider ces deux opérations ((4.9) et (4.10)) par l'observation du comportement d'un MPR, alors il peut considérer que sa relation de confiance envers ce MPR est correcte, telle qu'elle est exprimée dans (chapitre 3, page 31, expression 3.6).

Inversement, si (4.9) et (4.10) ne peuvent être validées pour un nœud sélectionné comme MPR, la relation de confiance est rompue et les nœuds l'ayant sélectionné comme MPR doivent se méfier de lui :

1. **Vérification de la génération de messages TC.** Si un nœud x a sélectionné y comme MPR, et ce dernier ne génère pas correctement les messages TC annonçant x , c'est-à-dire y ne génère pas de message TC ou génère un TC qui n'inclut pas x , alors y ne se comporte pas selon la spécification de OLSR et le nœud x doit se méfier de lui :

$$y \in MPRS_x, (x \xleftarrow{TC_y} y) \text{ ou } (x \xleftarrow{TC_y} y, x \notin MPRSS_y) \Rightarrow x \neg \text{trusts}(y) \quad (4.11)$$

2. **Vérification de la retransmission de paquets de données et de messages TC.** Si un nœud y sélectionné comme MPR ne retransmet pas les paquets de données ($DATA_x$) ni les messages (TC_x) envoyés par un nœud x qui l'a choisi comme MPR, alors y ne respecte pas la spécification de OLSR et x doit se méfier de lui :

$$y \in MPRS_x, (x \xrightarrow{TC_x} *, x \xleftarrow{(TC_x)_y} y) \text{ ou } (x \xrightarrow{DATA_x} *, x \xleftarrow{(DATA_x)_y} y) \Rightarrow x \neg \text{trusts}(y) \quad (4.12)$$

Ce contrôle peut être utilisé pour détecter les attaques visant à modifier la topologie par le biais de la fabrication ou de la modification des messages TC. Cette validation peut en particulier être utilisée contre les attaques de type trou noir (*blackhole attacks*), où un attaquant sélectionné comme MPR reçoit et élimine les messages de données au lieu de les retransmettre.

4.2.2.2 Validation de la sélection MPR des voisins

Suite à la sélection MPR, chaque nœud MPR doit jouer le rôle du routeur et ainsi recommander aux nœuds qui l'ont choisi comme MPR des chemins vers les nœuds distants (confiance exprimée par les relations (3.6, chapitre 3, page 31) et (3.7, chapitre 3, page 31)) :

$$\forall z \in MANET, \exists y \in MPRS_x : x \text{ trusts.rec}_{fw}(y) \text{ when.path}[\text{route}_{y \rightarrow z}] \text{ when.target}[z]$$

Cette expression présente la règle générale de la confiance dérivée pour le routage dans les réseaux fonctionnant sous le protocole OLSR. Il est donc important d'insister sur le contrôle et la validation des MPR car ils sont les premiers éléments de cette chaîne de confiance.

Pour vérifier le comportement de voisins, chaque nœud doit vérifier si ses voisins réalisent correctement la sélection de leurs MPR et le calcul de la table de routage, puis que les informations qu'ils annoncent sont cohérentes avec les informations observées localement. Selon l'algorithme de sélection de MPR (spécifié dans la RFC [CJ03]), si 2 nœuds, x et y , ont le même voisinage ($NS_x = NS_y$), ils devraient également sélectionner les mêmes MPR, sauf pour le cas où les différents MPR ont les mêmes voisinages :

$$\begin{aligned} NS_x - \{y\} = NS_y - \{x\} &\Rightarrow MPRS_x = MPRS_y \text{ et} \\ \forall z \in MPRS_x, \exists w \in MPRS_y : NS_z = NS_w & \end{aligned} \quad (4.13)$$

Cette propriété permet à x et y (et à n'importe quel voisin commun entre eux – $NS_x \cap NS_y$) de comparer la sélection de MPR entre x et y .

Comme nous l'avons vu dans le chapitre 3, le premier critère de sélection des MPR est le degré d'accessibilité. Chaque nœud x commence par sélectionner comme MPR le voisin y qui est le seul nœud capable d'offrir l'accessibilité/une route vers des voisins à deux sauts :

$$\begin{aligned} MPRS_x = MPRS_x \cup \{y \in NS_x : \exists z \in 2HNS_x \text{ tel que :} \\ z \in NS_y \text{ et } \forall w \in NS_x : z \notin NS_w\} \end{aligned} \quad (4.14)$$

Comme les liens symétriques des voisins ne sont pas vérifiés, un nœud malveillant peut annoncer une adresse supplémentaire inutilisée (z) et annoncer cette adresse avec le statut de lien symétrique de sorte que, sur réception de ce message HELLO, tous ses voisins le choisiront comme MPR sans aucune preuve de l'existence de ce lien.

Il est difficile de vérifier tous les statuts des liens de voisins. Mais, pour les nœuds choisis comme MPR, cette vérification est indispensable, compte tenu de leur rôle important comme routeurs. Dans OLSR, la confiance placée dans un MPR est plus importante que la confiance en un voisin symétrique [AdSJB07], ce qui implique que les MPR présentent un risque plus élevé s'ils sont compromis. Cela renforce la nécessité de vérifier continuellement les informations fournies par les nœuds MPR.

Il existe d'autres propriétés qui peuvent être vérifiées à la réception des messages TC d'un voisin, et qui permettent à un nœud à la fois de valider ses propres sélections de MPR et de vérifier les sélections de MPR effectuées par ses voisins :

- Les nœuds ayant le même voisinage (les mêmes voisins symétriques) ne doivent pas être sélectionnés comme MPR par le même nœud :

$$NS_x - \{y\} = NS_y - \{x\} \Rightarrow TC_x \neq TC_y, MPRSS_x \cap MPRSS_y = \emptyset$$

En effet, la spécification de OLSR définit une sélection de MPR optimisée. Par exemple, dans la figure 4.1, si un nœud z , voisin commun de x et de y ($z \in NS_x$ et $z \in NS_y$) sélectionne l'un d'eux comme MPR, par exemple x , alors il pourra accéder à tous ses voisins (NS_x) qui sont identiques à ceux de y (NS_y), et par conséquent, il n'a pas besoin de sélectionner y comme MPR car les voisins de y sont déjà accessibles. Dans le cas contraire, la situation représente une incohérence :

$$z \xrightarrow{HELLO_x} x, z \xrightarrow{HELLO_y} y, NS_x = NS_y, \exists w \in MPRSS_x \cap MPRSS_y \Rightarrow z \text{-trusts}(\{x, y, w\})$$

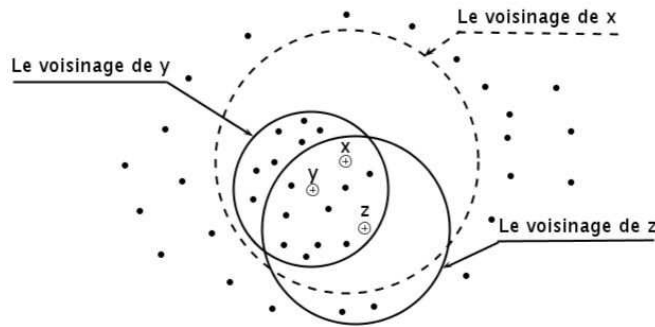


FIG. 4.1 – Exemple de deux nœuds x et y ayant le même voisinage

- Si un nœud y , ayant un voisinage inclus dans celui d'un autre nœud x , alors il ne peut pas être sélectionné comme MPR.

$$NS_y - \{x\} \subset NS_x - \{y\} \Rightarrow MPRSS_y = \emptyset$$

En effet, un nœud z , voisin de y et par conséquent voisin de x ($NS_y - \{x\} \subset NS_x - \{y\}, z \in NS_y \Rightarrow z \in NS_x$), vérifie le degré d'accessibilité de ses voisins avant de sélectionner ses MPR selon la règle 3.5 (chapitre 3, page 31), puisqu'il existe au moins le nœud x qui a un degré supérieur à celui de y ($D_x > D_y$), alors y ne sera pas sélectionné comme MPR (exemple dans la figure 4.2). Dans le cas contraire, si y a été sélectionné comme MPR et diffuse un message TC, alors cela représente une incohérence :

$$z \xleftarrow{HELLO_x} x, z \xleftarrow{HELLO_y} y, NS_x \subseteq NS_y, \exists w \in MPRSS_x \cap MPRSS_y \Rightarrow z \neg trusts(\{x, y, w\})$$

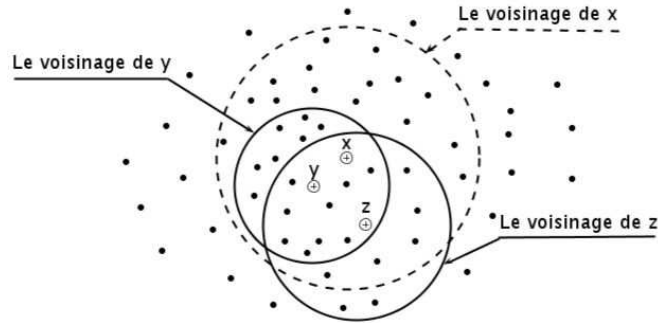


FIG. 4.2 – Exemple d'un nœud y ayant le voisinage inclus dans celui de x

Par conséquent, lorsqu'un nœud x détecte les incohérences dans la sélection MPR des voisins, il doit se méfier des nœuds concernés :

$$x \xleftarrow{HELLO_A} A, x \xleftarrow{HELLO_B} B, NS_A \subseteq NS_B, \exists z \in MPRSS_A \cap MPRSS_B \Rightarrow x \neg trusts(\{A, B, z\}) \quad (4.15)$$

Notons que x doit se méfier de A , de B et de z . Effectivement, à ce stade, x ne sait pas précisément qui est le nœud malveillant, il peut s'agir de z qui n'a pas choisi ses MPR correctement dans le but d'occuper les ressources de A et B , comme ça peut être A et B qui prétendent être MPR de z alors qu'ils ne devraient pas l'être. Une fois de plus, il convient de souligner que le raisonnement sur la méfiance ne peut pas toujours permettre l'identification précise du nœud à comportement anormal, mais permet de détecter une anomalie de comportement lié à un groupe de nœuds qui comprend l'attaquant.

4.2.3 Vérification de la cohérence de la topologie du réseau

Dans les réseaux ad-hoc, n'importe quel nœud peut être un routeur, et peut donc perturber le routage facilement en diffusant de fausses informations sur la topologie du réseau. Dans cet environnement, il n'y a aucune garantie que la route entre deux nœuds n'inclut pas un nœud malveillant. Il y a plusieurs attaques visant la topologie du réseau qui ont pour objectif de perturber le calcul de la table de routage des nœuds légitimes, et détourner le routage via des nœuds spécifiques.

Nous avons montré dans la section précédente que la découverte de voisinage et la sélection MPR peuvent être renforcées et validées en utilisant un raisonnement basé sur la confiance. Dans cette partie, nous présentons les propriétés de la table de routage, et montrons qu'un raisonnement basé sur la confiance peut être appliqué sur la table de routage pour valider les informations de la topologie du réseau.

4.2.3.1 La table routage dans OLSR

La table de routage est le résultat du protocole OLSR. Chaque nœud crée son point de vue de la topologie du réseau et calcule le plus court chemin vers toute destination en utilisant l'algorithme du plus court chemin de Dijkstra [Joh73]. La table de routage RT est décrite avec la formule suivante :

$$\forall z \in MANET, \exists y \in MPRS_x \Rightarrow \exists T \in RT_x, T = (z, y, N, I)$$

Chaque entrée de la table RT a le format suivant : (z, y, N, I) , où z est l'adresse destination se trouvant à N sauts via le nœud y , y est l'adresse du prochain voisin symétrique dans la route (qui doit être un MPR), et qui est accessible via l'interface local I .

Du point de vue de la confiance, chaque entrée dans la table de routage d'un nœud x , implique que x a confiance en y pour router vers z parce qu'il fournit le plus court chemin vers cette destination. Selon la définition de Dijkstra, le plus court chemin a deux propriétés :

1. Tout sous-chemin d'un plus court chemin est lui même un plus court chemin.

$$T = (B, y, N, I) \in RT_A \Rightarrow \forall x \in route_{A \rightarrow B} \exists t \in RT_x, t = (B, y_x, n_x, I), n_x < N \quad (4.16)$$

2. Dans un graphe valué, si $P_{A,B}$ est le poids du plus court chemin entre A et B , alors pour tout autre nœud x nous avons : $P_{A,B} \leq P_{A,x} + P_{x,B}$. Dans les réseaux ad-hoc, le poids de chaque lien est égal à 1, et donc la fonction du poids représente le nombre de sauts (distance). Ainsi, si $Dist(A, B)$ présente le nombre de sauts du plus court chemin entre A et B , alors :

$$\exists T = (B, x, N, I) \in RT_A \Rightarrow Dist(A, B) = N, \forall z \in MANET : Dist(A, B) \leq Dist(A, z) + Dist(z, B) \quad (4.17)$$

Pour le protocole OLSR, cette propriété signifie que si A sélectionne le nœud y comme MPR pour atteindre une destination B à N sauts, alors y doit choisir un MPR qui fournit un chemin vers la même destination avec $N - 1$ sauts.

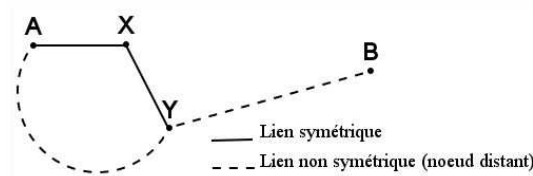


FIG. 4.3 – Exemple illustrant la propriété du plus court chemin

Cette propriété est représentée dans la figure (4.3). A choisit x comme MPR pour atteindre B (le plus court chemin) : $(B, x, N, I) \in RT_A$. x choisit y comme prochain saut MPR pour atteindre la même destination B : $(B, y, N - 1, I') \in RT_x$. Cette chaîne de sélection implique les deux points suivants :

1. y ne doit pas être voisin symétrique de A ($y \notin NS_A$). En effet, la distance entre y et la destination B est $N - 2$, donc si y est voisin symétrique de A , cela signifie que A s'est trompé dans le calcul des MPR et du plus court chemin, et qu'il aurait dû choisir y comme MPR et prochain saut pour atteindre cette destination parce qu'il fournit un chemin plus court que celui de x .
2. Quand x envoie ou retransmet un paquet de données à destination de B , alors ce paquet ne doit pas être retransmis par l'un des voisins de A , parce que les paquets de données ne sont relayés que par le nœud qui fournit le plus court chemin vers (B). Donc si le paquet est retransmis par un nœud z , $z \in NS_A \cap NS_x$, cela signifie que z fournit le plus court chemin à x (le MPR choisi par A), et que A s'est trompé et aurait dû le sélectionner comme MPR pour atteindre la destination de ce paquet de données.

Si l'une de ces deux situations est détectée, alors A s'est trompé dans le calcul du plus court chemin vers la destination B . Dans la suite de ce chapitre, nous allons démontrer, en se basant sur les propriétés du plus court chemin, qu'il est possible de détecter ces situations par un raisonnement sur la confiance dans le comportement des voisins.

4.2.3.2 Limitation du raisonnement sur la validation des MPR

Le raisonnement présenté précédemment pour la validation du calcul de MPR local et celui des voisins est utile pour palier à certaines vulnérabilités, et vérifier la cohérence des informations locales. Cependant, ce raisonnement n'est pas suffisant pour la détection de toutes les attaques contre OLSR, et surtout pour la détection d'incohérence dans le calcul de la table de routage. Pour cela, nous présentons un exemple d'attaque basée sur la modification de message HELLO et TC, qui a pour but de donner une fausse vision de la topologie du réseau en altérant le calcul de la table de routage, et qui n'est pas détecté avec le raisonnement de confiance sur les MPR.

Pour présenter cette attaque, il est important de souligner que, selon le protocole OLSR, chaque nœud est capable de calculer la table de routage de ses voisins, et donc d'avoir leur vision de la topologie du réseau :

Théorème 2. *Selon la spécification de OLSR [CJ03], chaque nœud peut détecter la vision de la topologie du réseau de ses voisins symétriques.*

$$\forall y \in NS_x : x \text{ peut calculer } RT_y$$

Démonstration. Supposons que x et y soient des voisins symétriques ($y \in NS_x$ et $x \in NS_y$).

La table de routage du nœud y est calculée en se basant sur les informations suivantes :

- $MPRS_y$: Puisque y doit annoncer tous ses voisins et leur type de lien (voisin MPR, voisin symétrique et voisin asymétrique) dans son message HELLO, alors son voisin x peut déduire facilement l'ensemble de ses MPR $MPRS_y$ après la réception de $HELLO_y$.

- TS_y : La table *topology set* du nœud y est construite à partir des informations reçues dans chaque message TC provenant du réseau. Le nœud y calcule sa table de routage à partir de TS_y . Puisque les messages TC sont diffusés dans tout le réseau, alors x va recevoir les mêmes messages TC que y , et il est capable de calculer la même table de topologie que y .

Par conséquent, x peut déduire et enregistrer l'ensemble des MPR et la table de topologie de n'importe quel voisin y , et donc peut calculer sa table de routage (RT_x). Ce résultat implique que x peut déduire aussi la distance entre y et n'importe quel nœud dans le réseau. □

En tenant compte de ce théorème, L'attaque sur la topologie du réseau (la table de routage) est définie selon les étapes suivantes (figure 4.4) :

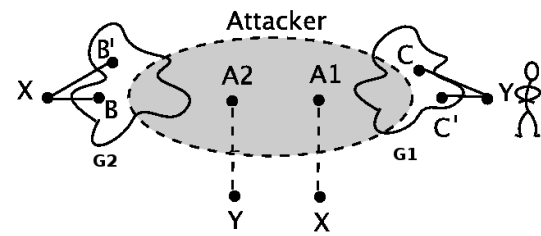


FIG. 4.4 – Modèle d'attaque sur la topologie du réseau

1. Soit $A \in MANET$ un nœud malveillant (attaquant). Au début, l'attaquant se comporte correctement pour collecter des informations sur le réseau et construire une vue correcte de la topologie du réseau.
2. L'attaquant choisit une cible x tel que $D(A, x) > 3$ (de préférence l'attaquant doit être placé le plus loin possible de sa cible), de tel sorte à pouvoir localiser les deux ensemble de nœuds $G1$ et $G2$ ayant les caractéristiques suivantes :
 - Selon le théorème 2, l'attaquant peut calculer la distance entre ses voisins et la cible x . Les nœuds de l'ensemble $G1$ sont les voisins qui sont plus éloignés de la cible que A :

$$G1 = \{z \in NS_A / D(z, x) \geq D(A, x)\}$$

- $G2$ est défini comme suit :

$$G2 = NS_A - G1$$

$$G1 \cap G2 = \emptyset \text{ et } G1 \cup G2 = NS_A$$

3. L'attaquant sélectionne sa deuxième cible y parmi l'ensemble des nœuds suivant :

$$G3 = \bigcap_{z \in G2} \{y / D(z, y) > D(y, A)\}$$

Si $G3 = \emptyset$, l'attaquant doit choisir une autre cible x ou bien changer sa position dans le réseau.

4. L'attaquant prend deux différentes identités $A1$ et $A2$ avec deux différentes interfaces (nous pouvons aussi supposer avoir deux nœuds attaquants travaillant ensemble).
5. Chacun de ces deux attaquants annonce l'autre comme son voisin symétrique et MPR, et modifie sa vision locale de tel sorte à partager le voisinage en deux parties :

$$\begin{aligned} NS_{A1} &= G1 \cup \{A2, x\}, & x, A2 &\in MPRSS_{A1}, \forall z \in G2, & (z, A2, N, I1) &\in RT_{A1} \\ NS_{A2} &= G2 \cup \{A1, y\}, & y, A1 &\in MPRSS_{A2}, \forall z' \in G1, & (z', A1, N, I2) &\in RT_{A2} \end{aligned}$$

6. Par conséquent, quand le message TC_{A1} est diffusé (avec $x \in TC_{A1}$) :
 - Quand $A2$ reçoit ce message, il génère un autre message TC n'incluant pas le nœud x (TC'_{A1}), et le diffuse comme étant le message TC de $A1$ (si l'attaquant est un seul nœud ayant deux identités, alors il génère deux messages TC différents) vers les nœuds de $G2$ et donc vers x . à la réception du message TC modifié TC'_{A1} , x ne va pas détecter une incohérence car il ne sera pas inclus dans ce message.
 - Quand les nœuds inclus dans $G1$ reçoivent TC_{A1} et $HELLO_{A1}$, ils vont tous sélectionner $A1$ comme MPR pour atteindre x , comme il semble offrir le plus court chemin vers x , et sera donc le prochain saut vers x dans leur table de routage :

$$\forall z \in G1, (x, A1, 2, I) \in RT_z \quad (4.18)$$

- De même, les nœuds inclus dans $G1$ et qui sont MPR de $A1$ vont retransmettre le message TC_{A1} , quand y reçoit ce message, il va mettre à jour sa table de routage avec la nouvelle distance calculée vers x .
7. De même, quand le message TC_{A2} est diffusé (avec $y \in TC_{A2}$) :
 - Quand $A1$ reçoit ce message, il génère un autre message TC n'incluant pas le nœud y (TC'_{A2}), et le diffuse comme étant le message TC de $A2$ vers les nœuds de $G1$ et donc vers y . A la réception du message TC modifié TC'_{A2} , y ne va pas détecter une incohérence car il ne sera pas inclus dans ce message.
 - Quand les nœuds inclus dans $G2$ reçoivent TC'_{A2} et $HELLO_{A2}$, ils vont tous sélectionner $A2$ comme MPR pour atteindre y , comme il semble offrir le plus court chemin vers y , et sera donc le prochain saut vers y dans leur table de routage :

$$\forall z \in G2, (y, A2, 2, I) \in RT_z \quad (4.19)$$

- De même, les nœuds inclus dans $G2$ et qui sont MPR de $A2$ vont retransmettre le message TC'_{A2} , quand x reçoit ce message, il va mettre à jour sa table de routage avec la nouvelle distance vers y .
8. Le nœud $A1$ va retransmettre tous les paquets provenant de $A2$ et $G2$ à destination de y , et le nœud $A2$ va retransmettre tous les paquets provenant de $A1$ et $G1$ à destination de x , afin de ne pas être détectés comme des attaquants réalisant un déni de service.

Il est important de noter que dans ce scénario d'attaque chaque paquet arrive à destination, mais pas forcément en suivant le plus court chemin. Le routage fonctionne correctement, mais l'attaquant a réussi à donner une fausse vision de la topologie du réseau, et a pu gagner la confiance de ses voisins qui l'ont choisi comme MPR pour router vers x suivant le plus court chemin. Cette situation donc représente une attaque aux propriétés de la confiance, l'attaquant n'ayant pas respecté la spécification de OLSR, mais ayant gagné la confiance de certains nœuds pour router via un chemin qui n'existe pas.

Cependant, il est important de souligner que ce scénario n'est pas applicable facilement, car l'attaquant doit être bien positionné dans le réseau pour pouvoir réussir et contourner le trafic entre les cibles (x et y) à travers lui même. Cependant, nous avons implémenté cette attaque dans un simulateur de réseau pour tester sa faisabilité, et nous avons pu vérifier que l'attaque est réalisable dans la majorité des cas.

4.2.3.3 Détection de l'attaque sur la topologie du réseau

Afin de détecter l'attaque sur le routage, nous présentons dans cette section les propriétés du plus court chemin sous l'optique du routage avec le protocole OLSR, et comment les utiliser pour détecter l'attaque. Nous supposons dans la suite de notre travail, que chaque nœud envoie une réponse après la réception d'un paquet de données ($DATA$).

Propriétés du plus court chemin Selon le principe de la sélection des MPR et le calcul de la table de routage dans OLSR [CJ03], nous avons déduit un théorème dérivé des propriétés du plus court chemin présentées dans les expressions (4.16) et (4.17). Ce théorème concerne l'unicité de la longueur du plus court chemin, c-à-d, si un nœud x calcule le plus court chemin vers y ayant une longueur de N sauts, alors y doit avoir la même longueur du plus court chemin vers x .

Théorème 3. *Soit x et y deux nœuds distants. Si x calcule le plus court chemin vers y ayant une longueur de N sauts, alors y doit avoir la même longueur du plus court chemin vers x .*

$$\begin{aligned} \exists t = (y, M_x, N, I_x) \in RT_x, D(x, A) = N \\ \text{and } \exists t' = (x, M_y, N', I_y) \in RT_A, D(A, x) = N' \Rightarrow N = N' \end{aligned} \quad (4.20)$$

Démonstration. $N \in \mathbb{N}^*$

Si $N = 1$, x et y sont des voisins symétriques $N = N' = 1$.

Si $N = 2$, x et y sont des voisins à 2 sauts $N = N' = 2$.

Si $N \geq 3$: soit $route_{x \rightarrow y}^N$ le plus court chemin calculé par x pour atteindre y ayant une longueur de N sauts. Puisque le chemin entre x et y est une succession de voisins symétriques, Alors il y a au moins un chemin entre y et x ayant la longueur de N sauts, et qui est le chemin opposé de la route calculé par x : $route_{x \rightarrow y}^N$.

Si y a choisi un autre plus court chemin $route_{y \rightarrow x}^{N'}$, alors :

- $N' < N$: alors le chemin calculé par y ($route_{y \rightarrow x}^{N'}$) est plus court que celui calculé par x , et que le chemin opposé de $route_{y \rightarrow x}^{N'}$ partant de x vers y , est plus court chemin calculé par x pour atteindre y , et donc on a une contradiction, $route_{x \rightarrow y}^N$ n'est pas un plus court chemin.
- $N' > N$: alors le chemin calculé par x ($route_{x \rightarrow y}^N$) est plus court que celui calculé par y , et que le chemin opposé de $route_{x \rightarrow y}^N$ partant de y vers x , est plus court que le chemin calculé par y pour atteindre x , on a donc contradiction aussi, $route_{y \rightarrow x}^{N'}$ n'est pas le plus court chemin.

Alors, $N = N'$, les deux plus court chemin ont la même longueur. \square

Notons que les deux chemins ne sont pas forcément les mêmes : $route_{x \rightarrow y}$ peut être différente de $route_{y \rightarrow x}$, mais ils doivent avoir la même longueur en nombre de sauts.

Détection d'attaque contre la topologie Supposons que x sélectionne le voisin B comme MPR et prochain saut pour atteindre la destination y ($(y, B, N, I) \in RT_x$), et que le nœud y sélectionne le MPR C comme prochain saut vers la destination x ($(x, C, N, I) \in RT_y$). Quand les routes sont calculées correctement, si le nœud x envoie un paquet de données vers y alors, selon le théorème 3, y doit recevoir ce paquet à partir de l'un de ses voisins qui fournit un chemin vers x ayant la même longueur que le plus court chemin calculé par y .

Supposons que l'attaque décrite à la section 4.2.3.2 est en cours. y calcule le plus court chemin vers x qui n'est pas le chemin correcte ($(x, C', N', I) \in RT_y$). Quand y envoie un message de données à x , ce paquet ne va pas suivre N' sauts comme supposé par y , mais il va suivre un chemin avec $D(y, A1) + D(A2, x) + 1$ sauts : car le paquet va passer de y à $A1$ ($D(y, A1)$ sauts), de $A1$ à $A2$ (1 saut) et de $A2$ à x ($D(A2, x)$). Quand le nœud destination x reçoit ce paquet, il vérifie le nombre de sauts N_p fournis dans le paquet reçu, et il en résulte deux cas possibles : $N_p = N$ ou $N_p \neq N$. En utilisant le raisonnement sur la confiance, nous allons montrer comment cette attaque est détectée, et prouver que les deux cibles x et y et les autres nœuds du réseau sont capables de détecter l'attaque.

Cas 1 : $N_p \neq N$

Si x détecte que $N_p \neq N$, l'attaque est détectée car cette situation ne correspond pas à la spécification de OLSR. En terme de confiance, x doit se méfier de tous les nœuds inclus dans la route qui a été choisis par y , car l'un d'entrer eux est l'attaquant qui tente de dévier les messages de y afin de les intercepter. De plus, x doit se méfier aussi du nœud y , parce qu'il peut être un attaquant qui n'a pas sélectionné ses MPR correctement pour donner une fausse vision de la topologie du réseau. Cette méfiance est résumée dans la formule suivante :

$$(y, B, N, I) \in RT_x, \exists B' \in NS_x, x \xleftarrow{DATA_{y-x}} B' : \\ hop_count(DATA_{y-x}) \neq N \Rightarrow \forall w \in route_{y \rightarrow B'}, x \neg trusts(w) \quad (4.21)$$

Où hop_count est la fonction qui fournit le nombre de sauts inscrits dans le message reçu.

Cette expression signifie que x envoie un paquet à destination de y via le MPR B qui fournit le plus court chemin à cette destination ayant N sauts, mais quand x reçoit un paquet provenant de y via un voisin B' , x vérifie si le nombre de sauts dans le paquet reçu ($hop_count(DATA_{y-x})$) est égal à la distance du plus court chemin calculé localement (N). Lorsque les deux distances sont différentes, x doit se méfier du chemin calculé par y , et donc se méfier de tous les nœuds w constituant ce chemin $w \in route_{y \rightarrow B'}$.

Dans cette situation, il est difficile pour x de détecter précisément les nœuds malveillants du chemin. Le raisonnement basé sur la confiance ne permet pas une détection précise, mais permet la détection d'une anomalie dans le routage et nous fournit le groupe de nœuds concernés, comme présenté dans l'expression 4.21. Avec le même raisonnement, quand y reçoit un paquet de données en provenance de x , ou une réponse de ses précédentes requêtes, lui aussi peut détecter la même anomalie selon l'expression suivante :

$$(x, C, N_p, I') \in RT_y, \exists C' \in NSy, y \xleftarrow{DATA_{x-y}} C' : \\ hop_count(DATA_{x-y}) \neq N_p \Rightarrow \forall w \in route_{x \rightarrow C'}, y \neg trusts(w) \quad (4.22)$$

Cette expression représente le comportement de y dans cette attaque (s'il n'est pas l'attaquant lui-même), lorsqu'il détecte la même incohérence que x , y va se méfier du chemin calculé par x , et donc de tous les nœuds w constituant le chemin le plus court entre x et C' ($route_{x \rightarrow C'}$).

Il est important de souligner que les nœuds x et y ne peuvent pas reconnaître l'ensemble des nœuds appartenant à la route choisie respectivement par y ($route_{y \rightarrow B'}$) et x ($route_{x \rightarrow C'}$) car ils ne sont pas voisins et aucun d'entre eux ne peut calculer la table de routage de l'autre. Par conséquent, il est impossible de détecter l'attaque précisément.

Dans l'attaque, les messages TC générés par $A1$ et $A2$ seront différents des messages retransmis respectivement par $A2$ et $A1$:

$$x \in TC_{A1} \text{ et } x \notin (TC'_{A1})_{A2} \Rightarrow TC_{A1} \neq (TC'_{A1})_{A2} \\ y \in TC_{A2} \text{ et } y \notin (TC'_{A2})_{A1} \Rightarrow TC_{A2} \neq (TC'_{A2})_{A1}$$

Cette situation représente une incohérence et elle est détectée selon la règle de confiance 4.4 (page 42). En effet, cette règle permet de vérifier l'intégrité du message TC en comparant le message d'origine avec le message retransmis.

Par conséquent, les nœuds qui vont recevoir les deux messages TC différents, ayant le même numéro de séquence et générés par le même nœud, vont détecter une incohérence et doivent se méfier de l'origine du message TC et de celui qui l'a retransmis.

$$\forall z \in MANET, \quad (z \xleftarrow{TC_{A1}} *, z \xleftarrow{(TC'_{A1})_{A2}} *) \Rightarrow z \neg trusts(\{A1, A2\}) \\ \text{ou} \quad (z \xleftarrow{TC_{A2}} *, z \xleftarrow{(TC'_{A2})_{A1}} *)$$

Par exemple, ils s'agit des nœuds appartenant au champs radio de $A1$ et $A2$. Dans le cas où l'attaquant prend deux identités différentes $A1$ et $A2$ avec deux interfaces différentes, ce sont tous les voisins de l'attaquant qui vont détecter l'attaque.

En générale, tous les nœuds du réseau qui vont recevoir le message TC généré par l'attaquant (TC_{A1} ou TC_{A2}) et le message généré par son complice (TC'_{A1} ou TC'_{A2}) vont détecter cette attaque selon la règle 4.4 selon la règle suivante :

$$\forall z, v, w \in MANET, (z \xleftarrow{(TC_{A1})_v} *, z \xleftarrow{(TC'_{A1})_w} *) \Rightarrow z \neg trusts(\{A1, v, w\})$$

$$\text{ou } (x \xleftarrow{(TC_{A2})_v} *, z \xleftarrow{(TC'_{A2})_w} *) \Rightarrow z \neg trusts(\{A2, v, w\}) \quad (4.23)$$

Cas 2 : $N_p = N$

Si x détecte que $N_p = N$, alors il y a deux possibilités :

- Le paquet est reçu via le voisin B' : x vérifie la longueur du chemin le plus court entre B' et y . S'il est différent de $N - 1$, alors une incohérence est détectée, sinon aucune attaque n'est détecté.
- Le paquet est reçu via le voisin B (et qui est le prochain saut vers y) : alors aucune incohérence n'est détectée,

Si $N_p = N$ et x ne peut pas détecter l'attaque, alors cela signifie que $A2$ a calculé correctement le nombre de sauts et l'a modifié dans le paquet de données reçu du nœud $A1$, avant de le diffuser aux nœuds du groupe $G2$, ainsi le nouveau nombre de sauts sera égal à $N - D(A2, x)$. Cependant, pour réussir ce cas, l'attaquant $A2$ doit connaître la longueur du plus court chemin entre x et y ce qui est possible uniquement pour les voisins de x et y . Il faut également que l'attaquant modifie le nombre de sauts dans le paquets de données de y sans être détecté. De plus, l'attaquant doit se positionner correctement dans le réseau, entre x et y , de manière à pouvoir vérifier la condition $N > D(A2, x)$, et ne pas être détecté selon la règle 4.23. Si toutes ses contraintes sont écartés, alors l'attaquant va réussir son attaque et ne va pas être détecté par notre raisonnement sur la tble de routage.

4.3 Illustration : attaque par fabrication de message HELLO

Cette section représente la suite de l'étude de l'exemple du chapitre 3 (page 35). Notre but est de montrer que les nœuds raisonnant sur la confiance sont capables de détecter cette attaque. Noter que dans le chapitre 6, il y'a plus de résultats concrets.

Rappelons les étapes du déroulement de l'attaque :

1. $att \xleftarrow{HELLO} B$: l'attaquant identifie les voisins de B (A et C) comme voisins symétriques (LS_B) ;
2. $att \xleftarrow{HELLO} A$: l'attaquant identifie B comme voisin MPR de A (LS_A) ;
3. l'attaquant fabrique un message HELLO annonçant les voisins symétriques de A et B :

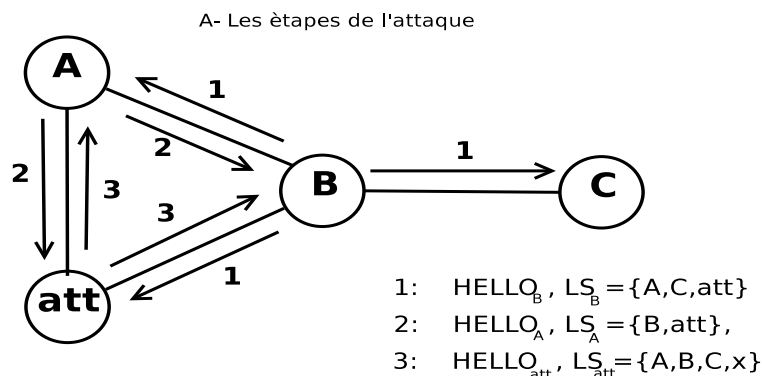


FIG. 4.5 – Fabrication de message HELLO

$$LS_{att} = LS_A \cup LS_B \cup x = \{(A, sym), (B, sym), (C, sym), (x, sym)\}$$

où x est un nœud fictif déclaré comme ayant un lien symétrique avec l'attaquant ;

4. l'attaquant doit également générer un message TC annonçant les nœuds x et C . En effet, puisqu'il est le voisin unique du nœud fictif x , ce dernier doit le choisir comme son MPR pour accéder à A , B et C , et le nœud C doit également le choisir comme MPR pour accéder à x : $MPRSS_{att} = \{x, C\}$

En conséquence de cette attaque, le nœud A va choisir l'attaquant att comme MPR unique pour router vers les nœuds x et C (selon la règle 3.4, chapitre 3, page 30), et va donc établir les relation de confiance exprimées par les formules suivantes :

$$\begin{aligned}
 &A \text{ trusts.rec}_{fw} (att) \text{ when.path}[route_{attaquant \rightarrow C}] \text{ when.target}[C] \\
 &A \text{ trusts.rec}_{fw} (att) \text{ when.path}[route_{attaquant \rightarrow x}] \text{ when.target}[x]
 \end{aligned}$$

Tandis que le nœud B va le choisir comme MPR pour router vers x , et établir la relation de confiance suivante :

$$B \text{ trusts.rec}_{fw} (att) \text{ when.path}[route_{attaquant \rightarrow x}] \text{ when.target}[x]$$

Une des possibles vérifications consiste à raisonner à partir des messages TC qui suivent. Avant l'attaque, A tenait B comme MPR pour router vers C et C tenait B comme MPR pour router vers A , on avait donc $MPRS_A \cap MPRS_C = \{B\}$. Après l'attaque, puisque B est toujours MPR de C , il va diffuser le message TC l'annonçant comme voisin. Par contre, l'attaquant va aussi diffuser un message TC annonçant A et B comme voisins.

Le raisonnement du point de vue de A va conduire à des conclusions contradictoires. En effet, après la réception du message TC de B , A déduit :

$$A \stackrel{TC_B}{\leftarrow} B, \{C\} \in MPRSS_B \Rightarrow B \in MPRS_C$$

Puisque les voisins de B sont les suivants : $\{A, C, att\}$, et l'attaquant est aussi voisin de C , alors A déduit que le nœud C a sélectionné B comme MPR pour router vers A ,

établissant la relation de confiance suivante :

$$B \in MPRSC \Rightarrow C \text{ trusts}_{fw}(B) \quad (4.24)$$

D'un autre coté, le nœud A , en recevant un message TC de l'attaquant, va déduire aussi les nœuds qui ont choisi ce dernier comme MPR :

$$A \xleftarrow{TC_{att}} att, MPRSS_{att} = \{A, B, C, x\} \Rightarrow \forall Z \in MPRSS_{att} : att \in MPRSZ$$

Pour vérifier la cohérence de ces messages TC, A compare entre les voisinages respectifs de B et de att , et va donc déduire que le voisinage de B est inclus dans celui de l'attaquant, et le degré d'accessibilité de B ($D(B)$) est inférieur à celui de l'attaquant. Ainsi, en se basant sur la clause 3.5 (page 31), A déduit que C devrait choisir l'attaquant comme MPR :

$$[NS_B - \{att\}] \subset [NS_{att} - \{B\}] \text{ et } D(B) < D(att) \Rightarrow att \in MPRSC \Rightarrow C \text{ trusts}_{fw}(att) \quad (4.25)$$

Selon la spécification de OLSR et la règle 4.15, ces déductions représentent une incohérence, et implique la relations de méfiance suivante :

$$A \xleftarrow{TC_B} *, A \xleftarrow{TC_{att}} *, C \in MPRSS_B \cap MPRSS_{att} \Rightarrow A \neg \text{trusts}(\{att, B, C\}) \quad (4.26)$$

Considérant que le nœud C ne devrait pas choisir B comme MPR, car son voisinage est accessible via le nœud att , il y a contradiction entre 4.24 et 4.25, ce qui mène le nœud A à se méfier des informations reçues. Le problème se pose au niveau de la sélection MPR du nœud C . Ainsi, le nœud A doit se méfier de ce nœud et des nœuds MPR qui sont concernés par l'incohérence. Notons que A ne détecte pas précisément le nœud malveillant.

Par ailleurs, le nœud B est capable de détecter l'attaquant selon les règles 4.6 et 4.8 en comparant le message $HELLO_C$ et le message TC_{att} :

$$B \xleftarrow{HELLO_C} C, B \xleftarrow{TC_{att}} *, \quad C \in NS_{att} \cap MPRSS_{att} \Rightarrow B \neg \text{trusts}(att) \\ , att \notin NS_C, att \notin MPRSC \quad (4.27)$$

De la même manière, Le nœud C est aussi capable de détecter l'attaque. Lorsqu'il reçoit le message TC de l'attaquant, il va détecter que ce dernier le déclare comme un sélecteur MPR alors qu'il n'est pas un voisin symétrique et par conséquent il n'est pas un nœud MPR :

$$C \xleftarrow{TC_{att}} *, C \in MPRSS_{att}, att \notin NS_C, att \notin MPRSC \Rightarrow C \neg \text{trusts}(att) \quad (4.28)$$

L'analyse de cet exemple d'attaque, sous l'optique de la confiance, indique l'importance de la corrélation d'informations reçues pour établir un contrôle basé sur la méfiance. Toutefois, certains nœuds ne sont pas capables de trouver précisément la source de l'incohérence détectée contrairement à d'autres nœuds qui arrivent à détecter précisément le nœud malveillant.

4.4 Conclusion

Pour palier à certaines vulnérabilités du protocole OLSR, nous avons proposé l'intégration d'un raisonnement basé sur la confiance dans chaque nœud du réseau. En effet, l'analyse réalisée a fait ressortir des possibles mesures pour rendre le protocole OLSR plus fiable et cela en exploitant des opérations et informations déjà existantes dans le protocole. Nous arrivons à la conclusion que des mécanismes de méfiance envers les comportements suspects peuvent être mis en place utilisant la corrélation entre les informations fournies dans les différents messages reçus. Par exemple, la découverte de voisinage, qui est limitée aux informations fournies par les messages HELLO, peut être renforcée en exploitant les informations topologiques (messages TC) pour valider les connaissances acquises et déduire d'autres critères qu'un nœud peut avoir pour sélectionner ses MPR.

La corrélation entre les messages reçus permet aux nœuds de valider leur vision locale et la vision de leur voisins. Lorsque les informations reçus sont cohérentes avec la spécification de OLSR, ce raisonnement permet la création et la validation de relations de confiance. Dans le cas contraire, il permet de se méfier des sources de ces incohérences (nœuds malveillants). De plus, nous avons montré que les messages de données (*DATA*) peuvent être également corrélés avec les informations fournies dans les messages HELLO et TC, et peuvent aussi fournir des informations sur la confiance dans le comportement des nœuds distants et ainsi permettre de vérifier la vision de la topologie du réseau. En effet, quand une incohérence est détectée par rapport à la spécification de OLSR, cette approche permet à chaque nœud de raisonner et de se méfier pas seulement de la vision local, mais aussi des nœuds distants et des routes concernées.

En conclusion, l'identification du processus de construction de la confiance entre les nœuds OLSR, déduite par l'analyse de la confiance implicite dans OLSR, nous a permis de créer un raisonnement basé sur des vérifications de cohérences des informations reçues. En intégrant ce raisonnement dans chaque nœud, il est possible d'évaluer le comportement des autres nœuds et de détecter les attaques contre le protocole OLSR, et donc de valider les relations de confiance entre les nœuds.

Cependant, ce raisonnement ne représente que l'étape de détection d'incohérence par rapport à la spécification de OLSR, et certains nœuds ne sont pas capables de détecter précisément la source de l'incohérence. De plus, aucune mesure n'est prise pour résoudre le cas d'une incohérence et contrer l'attaque. Il est donc important d'étudier les possibles contremesures qu'un nœud peut appliquer pour résoudre les problèmes détectés. De même, une simulation est nécessaire pour prouver la capacité et l'efficacité de la détection d'attaques.

Chapitre 5

Prévention et contremesures

L'intégration du raisonnement basé sur la confiance a permis à chaque nœud de vérifier la cohérence des informations reçues, et de détecter les nœuds malveillants. Dans ce chapitre, nous présentons deux mesures complémentaires pour répondre aux attaques contre OLSR : la prévention et les contremesures. La prévention permet de résoudre certaines vulnérabilités du protocole, et les contremesures traitent les comportements anormaux issus des vulnérabilités qui n'ont pas été résolues avec les mesures de prévention.

Dans notre travail, la prévention se base sur le mécanisme de signature des messages proposé par *M-SOLSR* [ACJ⁺03]. La différence est que nous utilisons le mécanisme d'*identité prouvable* au lieu d'une autorité de confiance centralisée sur laquelle se base *M-SOLSR* pour la certification des clés. Cette mesure résout la vulnérabilité de OLSR qui est due à l'absence de vérification des liens détectés au moment de la découverte de voisinage, mais ne résout pas d'autres vulnérabilités, comme par exemple l'absence de vérification du signalement des MPR et la surveillance des voisins MPR. Ainsi, lorsque d'autres vulnérabilités sont exploitées et une attaque est détectée, nous proposons des contremesures pour contrer et isoler les nœuds malveillants.

Il est à noter qu'il existe différentes attaques contre le routage OLSR, et que l'impact de ces attaques est différent d'un nœud à l'autre, car elles ne sont pas perçues et détectées de la même manière par tous les nœuds du réseau, ne mettant pas en cause les mêmes relations de confiance et ne représentant pas le même risque. En général, le risque que représente un nœud augmente avec la confiance qu'on lui accorde. Ainsi, l'impact d'une attaque dépend des relations de confiance ciblées par cette attaque. Par exemple, quand un nœud MPR est malveillant, il représente plus de risque comparé à un nœud symétrique malveillant, car la confiance accordée à un nœud MPR est plus importante. Par conséquent, les contremesures dépendent du type d'attaque et doivent être configurables et extensibles selon le domaine d'application et les besoins des utilisateurs. Dans ce chapitre, nous proposons des solutions qui peuvent être étendues selon le type d'attaque et les besoins des utilisateurs, et qui ne nécessitent pas la modification du protocole OLSR.

Nous commençons par la présentation du mécanisme de l'identité prouvable dans le

cadre du protocole OLSR. Ensuite, nous montrons comment l'identité prouvable peut être utilisé comme une méthode de prévention pour la validation de la découverte de voisinage et contre l'usurpation d'identité. Enfin, nous présentons les contremesures concernant les attaques contre les opérations de base de OLSR et une méthode de distribution des informations concernant la confiance pour prévenir les nœuds distants et permettre à tous les nœuds d'arriver à une détection précise.

5.1 Validation du voisinage avec l'identité prouvable

Dans OLSR, il est facile d'usurper l'identité d'un autre nœud, et difficile de vérifier si les liens déclarés par les voisins sont réels. Par exemple, dans l'attaque présentée dans la figure 4.5 (chapitre 4, section 4.3, page 55), il est impossible pour *A* de vérifier que le nœud *att* est vraiment voisin de *C* comme il le prétend. Pour pallier à ces vulnérabilités, nous nous basons dans notre travail sur la signature des messages.

Comme nous l'avons déjà signalé au chapitre 2, des versions sécurisées de OLSR se basant sur la signature des messages pour vérifier leur intégrité et leur authenticité ont été proposées [ACJ⁺03, HTR⁺04].

5.1.1 Présentation de *M-SOLSR*

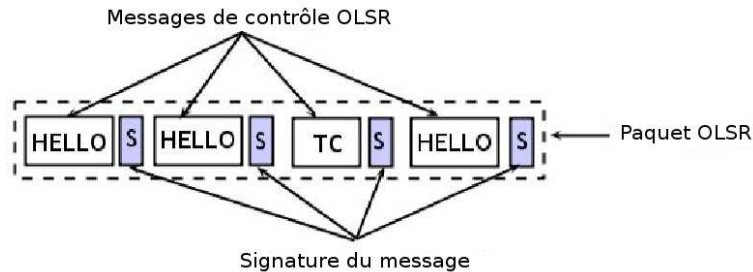


FIG. 5.1 – Format du paquet *M-SOLSR*

Dans *M-SOLSR* [ACJ⁺03], les auteurs définissent deux types de nœuds : des nœuds de confiance (*trusted nodes*) qui ne peuvent pas être compromis, et des nœuds inconnus non-sûrs (*untrusted nodes*). Ainsi, leur but est d'assurer que seul le trafic en provenance des nœuds de confiance est pris en compte, et que l'intégrité de ce trafic est garantie. Leur mécanisme se base sur la signature et l'horodatage de chaque message de contrôle (figure 5.1).

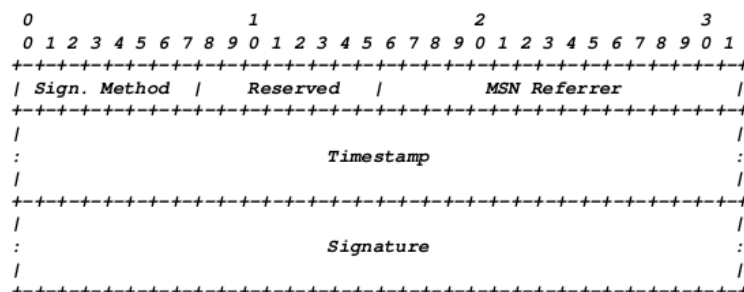


FIG. 5.2 – Format de la signature dans SOLSR

Pour empêcher des noeuds malveillants d’injecter des informations incorrectes dans le réseau, une signature est produite par le créateur de chaque message de contrôle et transmise avec le message de contrôle. En outre, un horodateur est associé à chaque signature, afin d’estimer la fraîcheur du message. Le format de la signature est présenté dans la figure 5.2.

Les signatures sont présentées comme un type séparé de messages de contrôle d’OLSR. Elles sont encapsulées et transmises comme un message dans le paquet OLSR décrit dans la figure 5.3.

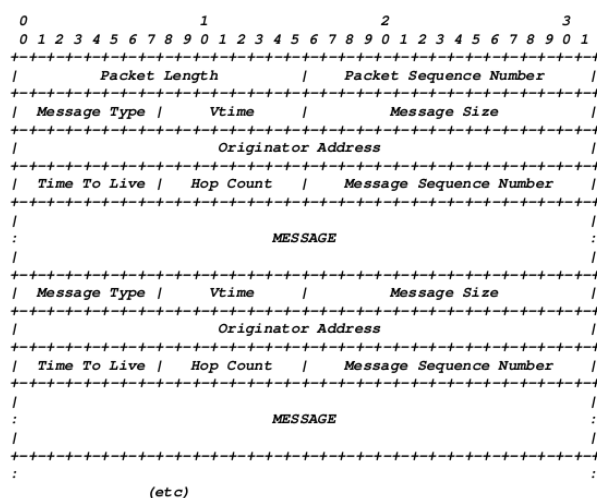


FIG. 5.3 – Format du paquet OLSR

En général, la signature et la vérification des messages dans SOLSR peuvent être effectuées avec une paire de clés privée et publique ou bien avec une clé secrète partagée¹. Dans ce dernier cas, la clé secrète partagée doit être distribuée d’une façon sécurisée afin qu’elle ne puisse être connue que par les noeuds de confiance.

¹Auquel cas, le mécanisme utilisé pour assurer l’intégrité est le calcul de MAC (Message Authentication Code).

Dans le cas de l'utilisation d'une paire de clés privée et publique, le problème réside dans la certification de la clé publique. Pour cela, les auteurs proposent deux solutions d'infrastructure à clé publique (*PKI*) : proactive et réactive. Les deux propositions utilisent une autorité de certification qui permet de définir les nœuds de confiance et de diffuser leur clés publiques. La différence est que l'infrastructure proactive vise la distribution des clés publiques dans le réseau, alors que dans la deuxième (réactive), les clés ne sont distribuées que sur demande des nœuds.

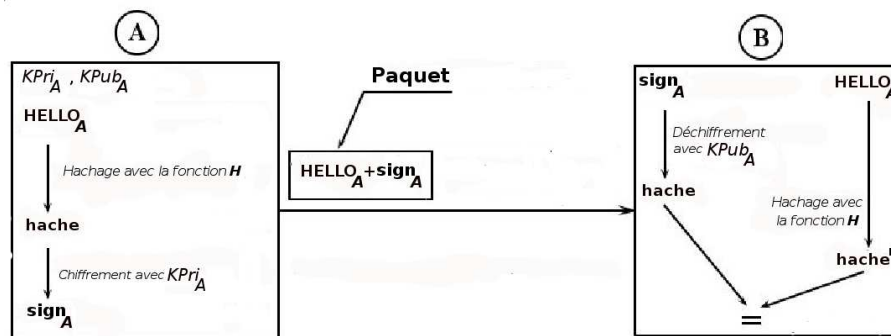


FIG. 5.4 – Exemple des échanges dans *M-SOLSR*

La figure 5.4 présente un exemple de signature et de vérification de message dans *M-SOLSR* en utilisant une clé publique. Avant l'envoi des messages HELLO, le nœud *A* crée le hache de son message HELLO à l'aide d'une fonction de hachage H (e.g. SHA-1). Ensuite, il le signe à l'aide de sa clé privée ($sign_A = [H(HELLO_A)]_{K_{Pri}_A}$). Enfin, il envoie avec le message HELLO, le hache signé ($sign_A$) dans un paquet. A la réception de ce paquet, le nœud *B* va procéder à la vérification de la signature du message pour s'assurer de l'intégrité des informations reçues. Étant donné que la fonction de hachage est un processus à sens unique (il est impossible de retrouver le message d'origine à partir du hache calculé), *B* doit comparer le hache calculé localement avec celui qui a été fourni par *A*. Il commence par déchiffrer $sign_A$ à l'aide de la clé publique K_{Pub}_A et déduit le hache du message HELLO calculé par *A* ($hache$), ensuite il calcule le hache du message $HELLO_A$ avec la fonction de hachage H et obtient $hache'$. Si les deux résultats sont identiques $hache = hache'$, alors *B* peut avoir la certitude que le message a été envoyé par le nœud *A*. Dans le cas contraire, si $hache \neq hache'$, alors le message a été altéré ou il n'a pas été signé par *A*, et dans ce cas *B* doit le rejeter.

Ainsi, à la réception d'un message de contrôle, il est possible de déterminer si le message provient d'un nœud de confiance, et si l'intégrité du message est préservée.

5.1.2 Présentation de *P-SOLSR*

Dans *P-SOLSR* [HTR⁺04], les auteurs proposent la signature et l'horodatage des paquets (et non pas de chaque message comme proposé dans *M-SOLSR* [ACJ⁺03]) en utilisant une clé secrète partagée². Les auteurs supposent que la clé secrète (symétrique)

²A nouveau, le mécanisme utilisé pour l'intégrité est donc le calcul d'un MAC.

soit sauvegardée dans un fichier déterminé (*/root/.solsrd/solsrd-key*) et qu'elle soit connue par tous les nœuds de confiance.

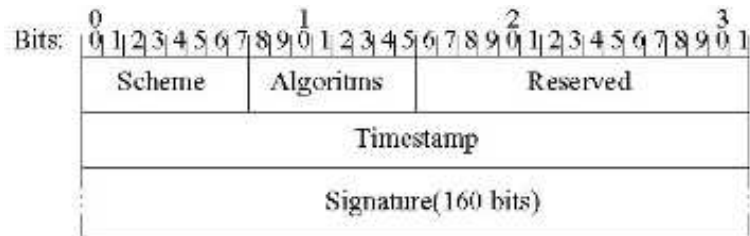


FIG. 5.5 – Format de la signature dans *P-SOLSR*

Le message de signature illustré dans 5.5 est attaché à chaque paquet OLSR, et il est le dernier message dans le paquet. Cela signifie qu'une seule signature est nécessaire pour le paquet OLSR, même s'il inclut plusieurs messages de contrôle (figure 5.6).

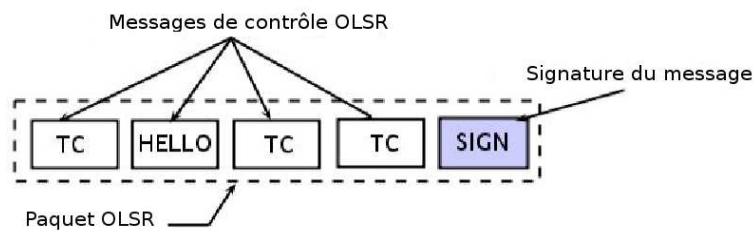


FIG. 5.6 – Format du paquet P-SOLSR

Dans *P-SOLSR*, la signature est un MAC (Message Authentication Code) calculé à l'aide d'une fonction de hachage (e.g., SHA-1) et de la clé secrète partagée, et portant sur les informations suivantes :

- l'entête du paquet OLSR,
- tous les messages de contrôle OLSR, excepté la signature du message.
- l'entête du message OLSR, la sous entête et l'horodatage de la signature du message.

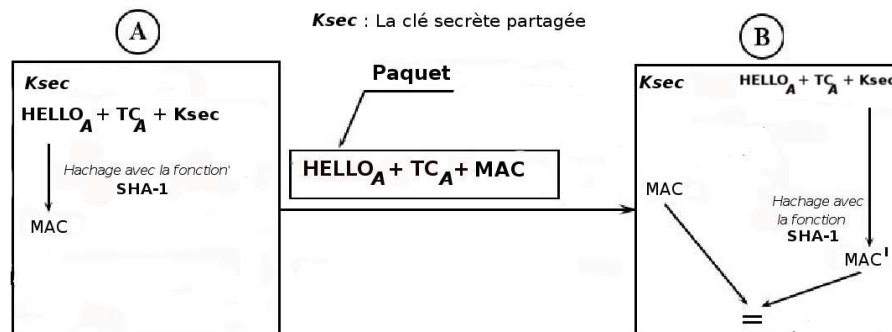


FIG. 5.7 – Exemple des échanges dans *P-SOLSR*

La figure 5.7 présente un exemple de signature de paquet (*P-SOLSR*). Avant l'envoi du paquet, le nœud *A* crée le hache de l'ensemble des messages du paquet et de la clé secrète K_{sec} à l'aide d'une fonction de hachage H , et obtient ainsi le MAC qu'il envoie dans le même paquet que les messages. A la réception de ces données, le nœud *B* va procéder à la vérification du paquet pour s'assurer de l'intégrité des informations reçues, ceci en calculant son MAC et le comparer avec celui qui a été fourni par *A* (MAC). Il commence par calculer le hache du paquet avec la fonction de hachage H et obtient MAC' . Si les deux résultats sont identiques $MAC = MAC'$, alors *B* peut avoir la certitude que le paquet a été envoyé par un nœud de confiance (ici, *A*), et l'ensemble des messages du paquet est accepté. Dans le cas contraire, si $MAC \neq MAC'$, alors le paquet a été altéré ou il n'a pas été envoyé par un nœud de confiance, et dans ce cas *B* doit le rejeter et aucun message n'est accepté.

Notons que, les auteurs ne proposent pas de solution basée sur l'utilisation des clés publiques. Ainsi, *P-SOLSR* est adapté seulement à un réseau ad-hoc constitué de nœuds de confiance se connaissant au préalable.

5.1.3 Aperçu de l'identité prouvable

La différence entre les deux solutions est dans les données signées. Dans *M-SOLSR*, les auteurs proposent de signer chaque message de contrôle OLSR. Tandis que dans *P-SOLSR*, les auteurs proposent de signer chaque paquet OLSR. La signature est encapsulée et transmise comme un message ordinaire d'OLSR. On déduit ainsi que les deux solutions n'impliquent que peu de changement dans OLSR. Contrairement à *P-SOLSR*, le message de signature dans *M-SOLSR* n'a pas besoin d'être dans le même paquet que le message signé. Ceci signifie que la signature et le message peuvent être envoyés dans des paquets séparés et suivant des itinéraires différents.

L'une des faiblesses de *P-SOLSR* est que la signature porte sur l'ensemble du paquet : lorsque le paquet OLSR contient un message TC qui a été retransmis par plusieurs nœuds, il est possible pour un nœud corrompu de modifier le contenu d'un message TC avant de générer la signature du nouveau paquet qui va le contenir. Par conséquent, *P-SOLSR* ne garantit pas l'authenticité et l'intégrité des messages TC.

La gestion des clés dans les deux solutions nécessite l'utilisation d'une autorité de certification (*M-SOLSR*) ou d'une clé secrète partagée (*P-SOLSR*). De notre point de vue, l'utilisation d'une entité centralisée ou de la cryptographie symétrique n'est pas adaptée à la philosophie des réseaux ad-hoc. Ainsi, nous utilisons dans notre travail la signature de message OLSR à base de cryptographie asymétrique, mais nous nous basons sur la notion d'identité prouvable afin d'éviter le besoin de certifier des clés publiques.

Nous appelons «identité prouvable» une identité qu'il est facile de vérifier, mais très difficile d'usurper. Le principe d'identité prouvable ne s'appuie pas sur une autorité centralisée pour prouver l'identité de chaque nœud du réseau. Le but n'est pas d'identifier les autres nœuds, mais une fois que l'identité de chacun est connue, elle nous assure que les messages suivants, venant de la même entité, seront authentiques, et qu'aucun autre nœud ne pourra usurper son identité.

En général, la clé publique d'une paire de clés publique/privée est utilisée comme identité prouvable : un nœud prétendant être identifié par sa clé publique peut signer un challenge en utilisant sa clé privée, et est le seul à pouvoir déchiffrer un message qui a été chiffré avec sa clé publique. Quelques exemples de mécanismes d'identité prouvable sont présentés par [MC02, OR01, LBPB06, PBAH03], utilisés notamment dans le cadre de la mobilité IPv6, et pour lesquels à chaque entité est associé le résumé cryptographique de sa clé publique.

L'identité prouvable est un mécanisme d'authentification adapté à la nature décentralisée des réseaux ad-hoc, car elle se base sur la cryptographie asymétrique et n'impose pas le partage de clé ou l'utilisation d'une autorité de certification. Elle permet à chaque nœud du réseau ad-hoc d'être identifié et de s'authentifier auprès des autres entités du réseau.

Actuellement, les protocoles utilisent l'adresse IP pour identifier les nœuds et supposent que chaque nœud utilise une adresse unique. Cependant, l'adresse IP ne suffit pas pour garantir l'identité de chaque nœud car elle peut être facilement falsifiée aboutissant à une usurpation d'identité, d'où le besoin de mettre un lien avec l'identité cryptographique (paire de clés publique/privée).

À l'IETF (Internet Engineering Task Force [IETF]), certains travaux proposent d'intégrer la notion d'identité prouvable dans IPv6 [DH98], afin de lier l'identifiant d'interface à la clé publique de l'émetteur du paquet.

5.1.3.1 Identité prouvable avec IPv6

Pour sécuriser les protocoles de découverte de voisins ou pour la gestion de la multi-domiciliation, certains travaux du groupe IETF proposent de lier l'identifiant d'interface (e.g. l'adresse IP) à la clé publique de l'émetteur du paquet. Plus précisément, ils suggèrent l'utilisation de **CGA** (*Cryptographically Generated Addresses* [Aur05]) pour la génération d'adresse. Pour chaque nœud, l'identifiant d'interface est généré en appliquant une fonction de hachage sur la clé publique et d'autres paramètres du nœud [DH98, ND01]. Ainsi, l'adresse IP ne peut pas être usurpée et constitue elle-même une

identité prouvable [Pal].

Dans IPv6, chaque nœud x ayant comme clé publique Pub_x , applique une fonction de hachage H sur sa clé publique (par exemple, la fonction SHA1) pour obtenir son adresse IP et en même temps son identité prouvable IdP_x :

$$@IP_x = IdP_x = H(Pub_x)$$

A la réception des messages de x , chaque nœud analyse leur intégrité en vérifiant la signature, et s'assure que le hache de la clé publique utilisée pour cette vérification est égale à l'adresse IP de x . Si les deux résultats sont égaux, alors l'émetteur x a pu prouver son identité, et le destinataire a pu valider l'authenticité des messages.

Par conséquent, si OLSR est appliqué au dessus de IPv6 alors l'identité prouvable peut être obtenue simplement en intégrant l'extension proposée par l'IETF [LBMA04]. Dans le cas contraire (OLSR sur IPv4), nous présentons dans le paragraphe suivant une approche permettant d'avoir de l'identité prouvable.

5.1.3.2 Identité prouvable pour OLSR avec IPv4

Dans OLSR, l'identité d'un nœud est équivalente à son adresse IP (RFC3626 [CJ03], section 20.4, page 70), et chaque nœud est supposé avoir une adresse unique. Ainsi, un contrôle d'identité plus robuste est nécessaire pour éviter les attaques d'usurpation d'identité et garantir que chaque message envoyé par une adresse IP ne puisse être modifié ou falsifié. Dans cette section, nous proposons un mécanisme d'identité prouvable adapté à OLSR où l'identité d'un nœud est associée à sa clé publique.

Notre approche se base sur la signature des messages OLSR (*M-SOLSR*). Nous exigeons cependant que la signature du message et la clé publique correspondante soient jointes au même paquet de ce dernier. La signature et la clé publique du nœud sont ajoutées au paquet OLSR comme un type particulier de message.

L'identité prouvable d'une entité est représentée sous la forme du hache de sa clé publique. Au début des échanges, chaque entité A envoie sa clé publique K_{Pub_A} aux autres entités qui vont stocker le hache de cette clé publique comme étant l'identité prouvable IdP_A de A . Au moment des échanges, chaque entité A signe ses messages à l'aide de sa clé privée K_{Pri_A} avant de les envoyer, et lorsque ces messages sont reçus, il est possible de vérifier leur intégrité et leur authenticité en vérifiant leur signature avec la clé publique de l'émetteur.

Ainsi, après la vérification de l'intégrité du message fournie, l'identité prouvable de l'émetteur va prendre la forme suivante :

$$IdP_A = (H(K_{Pub_A})),$$

Par conséquent, si deux nœuds utilisent la même adresse IP, ils sont considérés comme deux identités différentes, car ils ne possèdent pas la même clé publique, et l'un d'entre eux doit changer son adresse IP pour participer au réseau. Pour éviter cette situation, et garantir qu'une adresse IP ne soit utilisée que par un seul nœud, nous proposons l'utilisation d'une table *TabIP* indexée sur les adresses IP et qui associe une clé publique pour chaque adresse IP.

$$K_{Pub_A} = TabIP(@IP_A) \quad (5.1)$$

@IP	clé publique
A	K_{Pub_A}
B	K_{Pub_B}
C	K_{Pub_C}
.	.
.	.

TAB. 5.1 – Structure de la table d’adresses IP *TabIP*

Ainsi, si un nœud x ayant comme clé publique K_{Pub_x} possède la même adresse IP qu’un autre nœud A , la table *TabIP* va retourner la clé publique de A ($TabIP(@IP_A) = K_{Pub_A}$) qui a déjà été enregistrée et qui est différente de la clé publique de x ($K_{Pub_x} \neq K_{Pub_A}$). Alors les messages de x seront refusés et il devra modifier son adresse IP pour pouvoir s’intégrer au réseau. En particulier, si un nœud détecte qu’un autre nœud utilise la même adresse IP, alors il doit changer son adresse.

Si un nœud x change d’adresse IP sans changer sa clé publique, il sera reconnu sous une autre identité, car aucune clé publique n’a été enregistrée pour la nouvelle adresse IP. Ceci implique que la vérification de l’identité dans la table d’adresses IP n’est pas suffisante et il est important de vérifier aussi si son identité prouvable (hache de sa clé publique) a déjà été détectée. Pour cela, nous proposons l’utilisation d’une deuxième table, appelée table de hachage *TabH* 5.2, qui soit indexée sur le hache de la clé publique, et représente ainsi une optimisation car elle permet de raisonner sur le hache et non pas sur la clé publique :

$$@IP_A = TabH(H(K_{Pub_A}))$$

hache de la clé publique	@IP
H(K_{Pub_A)}	A
H(K_{Pub_B)}	B
H(K_{Pub_C)}	C
.	.
.	.

TAB. 5.2 – Structure de la table de hachage *TabH*

Dans ce cas, si un nœud ayant la même clé publique change d’adresse IP, alors sa nouvelle adresse IP est mise à jour dans les tables en gardant la même identité prouvable.

En résumé, la vérification d’adresse IP dans la table *TabIP* et des clés publiques dans la table *TabH* garantit qu’une adresse IP ne peut être utilisée que par un seul nœud, et si un nœud change d’adresse IP sans changer sa clé publique il sera détecté

avec la même identité prouvable. Ainsi, l'utilisation des deux tables $TabH$ et $TabIP$ nous permet de représenter l'identité prouvable sous la forme du hache de la clé publique 5.2 :

$$K_{Pub_A} = TabIP(@IP_A) \text{ et } @IP_A = TabH(H(K_{Pub_A})) \Leftrightarrow IdP_A = H(K_{Pub_A}) \quad (5.2)$$

Cette représentation de l'identité prouvable n'implique aucun changement sur la structure des messages et la table de routage dans OLSR. En effet, le mécanisme d'identité prouvable assure qu'une adresse IP est unique et ne peut être usurpée.

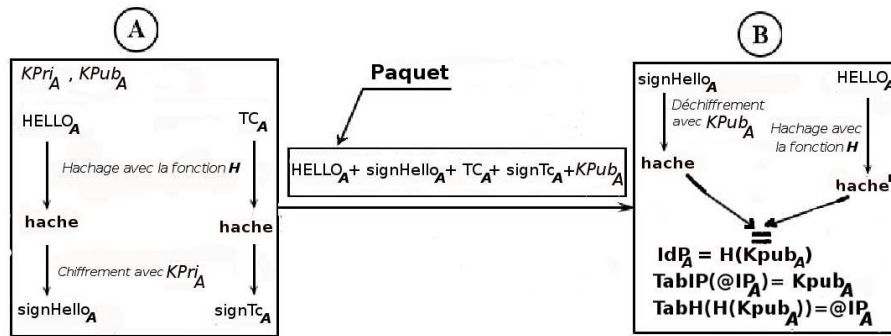


FIG. 5.8 – Exemple des échanges avec l'identité prouvable et $M-SOLSR$

La figure 5.8 représente un exemple d'échange dans OLSR en se basant sur l'identité prouvable. Dans cet exemple, B va détecter et valider l'identité de A à partir de la clé publique K_{Pub_A} fournie dans le paquet. Lorsque les messages envoyés sont valides, l'identité prouvable de A est enregistrée :

$$\begin{aligned} IdP_A &= H(K_{Pub_A}) \\ TabIP_B(@IP_A) &= K_{Pub_A} \\ TabH_B(H(K_{Pub_A})) &= @IP_A \end{aligned}$$

Ainsi, même si A envoie un paquet avec des messages différents, son identité restera la même car elle est associée à la même clé publique et la même adresse IP.

Par ailleurs, si un autre nœud prétend être le nœud A (en utilisant la même adresse IP et une clé publique différente), ses messages vont être refusés et il doit changer d'adresse IP. Par conséquent, si un nœud tente d'usurper l'identité de A en envoyant un paquet avec la clé publique de A ($HELLO_A + msg'_A$), la signature ne va pas concorder avec la clé fournie car elle est calculée avec une clé privée différente : $msg'_A = [H(HELLO_A)]_{K_{Pri}'_A}$. Ainsi, la vérification du hache va impliquer le rejet du message :

$$H(HELLO_A) \neq [msg'_A]_{K_{Pub_A}} \Rightarrow K_{Pri}'_A \neq K_{Pri}_A$$

En se basant sur la signature des messages (la solution proposée par $M-SOLSR$) et l'identité prouvable, nous proposons une méthode de prévention afin de garantir l'étape de découverte de voisinage. Cette méthode permet d'assurer l'intégrité des informations fournies par les voisins symétriques, et ainsi de valider la découverte des voisins (à un et à deux sauts) et la sélection MPR.

5.1.4 Preuve de voisinage

Dans cette section, nous montrons qu'il est possible de valider les informations concernant la découverte de voisinage en utilisant l'identité prouvable. Le but est de permettre à chaque nœud de fournir une preuve de l'existence d'un lien particulier (**preuve de voisinage**), et de vérifier les preuves de voisinage fournies par les voisins.

Après la réception d'un paquet, chaque nœud détecte l'identité de l'émetteur. Par exemple, dans la figure 5.8, le nœud B détecte l'identité de son voisin A à partir du paquet $P_A = HELLO_A + signHello_A + K_{pub_A}$ sous la forme suivante :

$$\begin{aligned} \text{si } P_A \text{ est valide} &\Rightarrow TabIP(@IP_A) = K_{Pub_A} \\ &TabH(H(K_{Pub_A})) = @IP_A \\ &IpP_A = H(K_{Pub_A}) \end{aligned}$$

Pour prouver à ses voisins qu'il est bien le voisin symétrique de A , nous proposons pour B de transmettre un paquet $Preuve_B$ incluant le message HELLO signé de A :

$$Preuve_B \leftarrow Preuve_B \cup \{HELLO_A + signHello_A + K_{Pub_A}\}.$$

Ainsi, tout voisin de B va vérifier que A est effectivement voisin de B et va déclarer A comme un voisin à 2 sauts. D'un autre côté, A pourra vérifier les liens de B avec ses autres voisins.

Après l'établissement d'un nouveau lien symétrique, la preuve n'est envoyée qu'une seule fois³, pour prouver la validité du nouveau lien symétrique établi aux autres voisins, et pour prouver la validité des autres liens symétriques au nouveau voisin.

Précédemment, nous avons raisonné sur l'identité des nœuds comme étant l'adresse IP, et nous avons supposé que cette identité ne peut pas être usurpée. Cette hypothèse est validée grâce à l'utilisation de l'identité prouvable. Maintenant, pour les ensembles : LS , NS , $2HNS$, $MPRS$, $MPRSS$, RT , TS , nous allons raisonner sur l'identité prouvable. Mais pour les messages HELLO et TC nous raisonnons sur l'adresse IP afin que le protocole OLSR reste inchangé.

Par exemple, en utilisant l'identité prouvable dans la formule 4.2 (chapitre 4, section 4.2.1.1, page 41), on obtient :

$$x \xleftarrow{TC_y} *, @IP_x \in TC_y, IdP_y \notin NS_x \Rightarrow x \text{-trusts}(IdP_y) \quad (5.3)$$

Dans l'annexe A, nous présentons toutes les formules du raisonnement sur la confiance en utilisant le mécanisme d'identité prouvable.

Après la réception de $Preuve_B$, tous les voisins symétriques de B vont procéder à la vérification suivante ($\forall(z, sym) \in LS_B$) :

³Le paquet $Preuve_B$ n'est envoyé que par le nœud B , et il ne doit pas être retransmis : ce paquet ne sera pas diffusé dans tout le réseau.

- Vérification de l'intégrité du paquet $Preuve_B$: le nœud z doit d'abord vérifier l'intégrité de chaque message. Par exemple, pour le message $HELLO_A$, il vérifie la concordance du hache avec la signature du $signHello_A$ en utilisant K_{Pub_A} . Si le message est validé alors z passe à l'étape suivante. Dans le cas contraire, le message est rejeté et il est impossible de prouver que B est réellement voisin de A , alors le nœud A ne sera pas considéré comme voisin à deux sauts.
- Vérification du lien symétrique : le nœud z procède ensuite à la vérification du lien symétrique pour s'assurer que le nœud A déclare aussi B comme un voisin symétrique :
 1. Si B est déclaré comme voisin symétrique dans le message HELLO de A ($HELLO_A \in Preuve_B$ et $(@IP_B, sym) \in HELLO_A$ ou $(@IP_B, mpr) \in HELLO_A$), alors le lien symétrique entre A et B est validé, et le nœud A est inséré dans l'ensemble des voisins à deux sauts de z .
 2. Sinon, le lien n'est pas validé, et les deux nœuds A et B sont déclarés comme malveillants selon la règle 4.8 (chapitre 4, page 43) en établissant la relation de méfiance suivante :

$$\begin{aligned}
 & z \xleftarrow{HELLO_B} B, [(@IP_A, sym) \in HELLO_B \text{ ou } (@IP_A, mpr) \in HELLO_B] \\
 & z \xleftarrow{Preuve_B} B, [(@IP_B, sym) \notin HELLO_A \text{ et } (@IP_B, mpr) \notin HELLO_A] \\
 & \quad \downarrow \\
 & z \neg trusts(IdP_A, IdP_B) \tag{5.4}
 \end{aligned}$$

Avec ce mécanisme, la réception des messages HELLO ne suffit pas pour détecter les voisins à deux sauts. Pour tout lien symétrique entre des nœuds x et y , chaque nœud z voisin de x doit recevoir une preuve valide $Preuve_x$ incluant $HELLO_y$ et déclarant x également comme voisin symétrique. Ainsi, la découverte de voisinage exprimée dans la règle de confiance 3.3 (chapitre 3, page 30) devient :

$$\begin{aligned}
 & \forall IdP_x \in NS_z, \forall y : (@IP_y, sym) \in HELLO_x \text{ ou } (@IP_y, mpr) \in HELLO_x : \\
 & z \xleftarrow{Preuve_x} x, (@IP_x, sym) \in HELLO_y \text{ ou } (@IP_x, mpr) \in HELLO_y \\
 & \quad \downarrow \\
 & z trusts_{id \cup ni}(IdP_x), 2HNS_z \leftarrow 2HNS_z \cup \{IdP_y\} \tag{5.5}
 \end{aligned}$$

En d'autres termes, ne sont insérés que les nœuds dont on possède la preuve de voisinage, permettant ainsi d'obtenir *un voisinage à deux sauts de confiance*. Par conséquent, les attaques visant le voisinage à deux sauts deviennent inefficaces car un nœud ne peut plus mentir à propos de ces voisins (les nœuds inclus dans le voisinage de confiance). Ce mécanisme de prévention va couvrir les règles de confiance concernant la vision locale, et va permettre une détection plus rapide concernant la cohérence des liens.

5.2 Contremesures

La première contremesure que nous proposons concerne les opérations de base de OLSR qui sont la découverte de voisinage et la sélection MPR, tandis que la deux-

ième concerne la distribution des informations de confiance, et la preuve de détection d'attaque pour prévenir les nœuds distants.

Dans les deux solutions, nous supposons que la technique d'horodatage proposée dans *M-SOLSR* [ACJ⁺03] et le mécanisme d'identité prouvable présentés précédemment sont mis en place afin d'assurer respectivement la fraîcheur et la validité des messages.

5.2.1 Sélection efficace des voisins et des MPR

Dans OLSR, lorsqu'un nœud détecte un voisin symétrique, il considère que les informations fournies par ce dernier sont fiables. Ainsi, il établit la relation de confiance 3.2 (chapitre 3, page 30) :

$$\begin{aligned}
 x \xleftarrow{HELLO_y} y, (@IP_x, sym) \in HELLO_y \text{ ou } (@IP_x, mpr) \in HELLO_y \\
 \Downarrow \\
 x \text{ trusts}_{id \cup ni}(IdP_y), LS_x \leftarrow LS_x \cup \{(@IdP_y, sym)\}
 \end{aligned}$$

En utilisant le raisonnement sur la confiance, lorsqu'un voisin symétrique est détecté comme étant malveillant, et ne respectant pas la spécification de OLSR, la relation de confiance 3.2 est rompue. Comme contremesure, nous proposons de rejeter les messages de ce voisin et de casser son lien symétrique. Ainsi, il sera isolé et les informations qu'il émet seront ignorées :

$$\begin{aligned}
 \forall IdP_y \in NS_{x, x-trusts}(y) \Rightarrow \quad & LS_x = LS_x - \{(IdP_y, sym)\}, \\
 & NS_x = NS_x - \{IdP_y\}, \\
 & MPRSS_x = MPRSS_x - \{IdP_y\}, \\
 & MN_x = MN_x \cup \{IdP_y\}, \\
 & \text{rejeter } HELLO_y \text{ et } TC_y, \quad (5.6)
 \end{aligned}$$

où MN_x (Mistrusted Nodes) représente l'ensemble des nœuds malveillants détectés par x . A chaque fois qu'une incohérence est détectée, l'identité prouvable du nœud malveillant concerné est incluse dans cet ensemble. Ainsi, même si le nœud malveillant change d'adresse IP, il sera toujours isolé.

Cette contremesure n'est prise que lorsque la détection est précise, et elle peut être permanente ou temporaire selon les besoins des utilisateurs et le contexte d'utilisation. Avec un rejet permanent le nœud malveillant sera ignoré de façon définitive et ne sera jamais accepté comme voisin symétrique, sauf s'il change d'identité, i.e. de clé publique. Tandis qu'avec un rejet temporaire, il sera ignoré pendant une durée déterminée, dépendant du contexte d'utilisation et de l'application.

Après la découverte de voisinage, chaque nœud sélectionne ses MPR parmi ses voisins symétriques, et cette sélection permet la construction d'une nouvelle relation de confiance avec les voisins MPR. En effet, lorsqu'un MPR est un nœud malveillant, il peut

contrôler le trafic des nœuds qui l'ont sélectionnés comme MPR, et pourra donner une fausse vision de la topologie du réseau.

Ainsi, lorsqu'un MPR est détecté comme malveillant, la première contremesure à prendre est la même contremesure que lorsqu'un voisin symétrique est malveillant (5.6) afin de casser le lien symétrique. En second lieu, il faut refaire la sélection MPR dans le but d'avoir un nouveau ensemble de MPR n'incluant pas le nœud malveillant. Il est important de souligner que si un nœud change seulement la sélection MPR en supprimant le nœud malveillant sans casser le lien symétrique, il sera lui même détecté comme incohérent et donc comme malveillant par ses voisins, car sa sélection de MPR, par rapport aux voisins symétriques qu'il annonce, sera incohérente et non conforme à la spécification de OLSR.

$$\begin{aligned}
 \forall IdP_y \in MPRS_{x, x \rightarrow trusts}(y) \Rightarrow & \quad LS_x = LS_x - \{(IdP_y, sym)\}, & (5.7) \\
 & \quad NS_x = NS_x - \{IdP_y\}, \\
 & \quad MPRSS_x = MPRSS_x - \{IdP_y\}, \\
 & \quad MN_x = MN_x \cup \{IdP_y\}, \\
 & \quad \text{rejeter HELLO}_y \text{ et TC}_y, \\
 & \quad \text{et refaire la sélection MPR du nœud } x.
 \end{aligned}$$

Il existe des cas où le voisin malveillant est le seul MPR à fournir l'accessibilité vers certains nœuds du réseaux, et le fait de casser le lien symétrique avec ce voisin signifie que le nœud ne pourra pas accéder à l'ensemble du réseau accessible à travers le voisin malveillant. Il est possible d'envisager de changer le calcul des MPR et de la table de routage, de manière à choisir le voisin malveillant comme MPR seulement pour router vers les nœuds qui ne sont accessibles qu'à travers celui-ci. Cependant, ce choix implique un routage multi-chemin [CH07], et une sélection MPR différente à celle du protocole OLSR. Cette solution n'est donc pas conforme à la spécification de OLSR de base et n'est pas compatible avec le raisonnement sur la confiance (qui se base sur la sélection MPR dans la spécification de OLSR de base). Par conséquent, nous pensons que la meilleure solution et contremesure à prendre est de casser le lien symétrique et ignorer tous ses messages, car quand un voisin symétrique ou MPR est malveillant, il n'existe aucune garantie sur l'existence des liens qu'il déclare.

5.2.2 Distribution de la preuve de méfiance

La deuxième contremesure que nous proposons est le partage d'informations relatives à l'attaque. Ce partage se fait par la diffusion d'une alerte afin de prévenir tous les nœuds du réseau de se méfier de l'attaquant.

Cependant, pour éviter les fausses alertes (faux témoignages), l'alerte doit fournir une preuve fiable de l'attaque qui ne peut pas être falsifiée. Lorsque la détection est précise, nous proposons de retransmettre les messages de contrôle qui ont permis de découvrir l'incohérence et de détecter l'attaque. D'un autre coté, l'identité prouvable et la signature des messages (*M-SOLSR*) assurent l'authenticité et l'intégrité de chaque

message, et garantissent ainsi la fiabilité et l'intégrité des alertes émises. Il est important de noter que l'alerte n'est pas un nouveau de type de message, mais elle représente la rediffusion des messages de contrôle qui ont révélé une incohérence.

Le raisonnement sur la confiance, pour la détection d'incohérence, se base sur la comparaison des informations fournies dans les messages reçus (HELLO, TC et DATA) avec la vision locale. Par conséquent, l'alerte se compose de deux messages : il s'agit des messages comparés pendant la phase de détection. L'alerte se fait par la retransmission des deux messages qui ont déclenchés l'incohérence (y compris les messages HELLO) et ont permis de détecter l'attaque selon deux scénarios :

1. Dans le cas où un nœud détecte une incohérence entre un message reçu et sa vision locale, il doit alerter les autres nœuds en retransmettant le message reçu, et le message qui représente sa vision locale (HELLO ou TC). Par exemple, si un nœud détecte une incohérence entre sa sélection MPR et le message reçu, alors il doit retransmettre le message reçu et son message HELLO, car le message HELLO va présenter sa sélection de MPR (figure 5.9). Par contre, si l'incohérence concerne l'ensemble des sélecteurs MPR, alors il doit retransmettre son message TC au lieu du message HELLO.

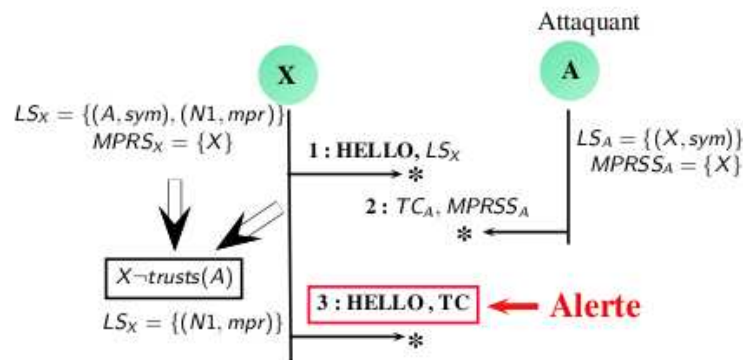


FIG. 5.9 – Exemple de détection d'attaque et envoi d'une preuve

2. Dans le cas où un nœud détecte une incohérence entre deux messages en provenance d'autres nœuds, alors il doit alerter tout le réseau en retransmettant ces messages. Par exemple : si un nœud détecte que son voisin x annonce des nœuds dans son message TC, mais ne les déclare pas dans ses message HELLO, alors il va détecter une incohérence entre les messages HELLO et TC de x reçu, et il doit les retransmettre.

Il est important de noter qu'il faut d'abord mettre en place la première contremesure (définie dans la section précédente) avant de transmettre les messages d'alerte. Précisément dans le cas où l'attaquant est lui même MPR, il faut casser le lien symétrique et refaire la sélection MPR pour garantir que l'alerte arrive à tous les nœuds du réseau. D'un autre côté, même si les messages TC sont censés arriver à tous les nœuds du réseau, quand l'attaquant est un MPR il peut perturber le routage, et donc il n'est pas

garanti que ces messages arrivent à tous les nœuds du réseau. Ainsi, quand l'alerte est constituée de messages TC et concerne un attaquant qui est aussi MPR, il est important de retransmettre ces messages après la détection de l'attaque et le changement de la sélection MPR.

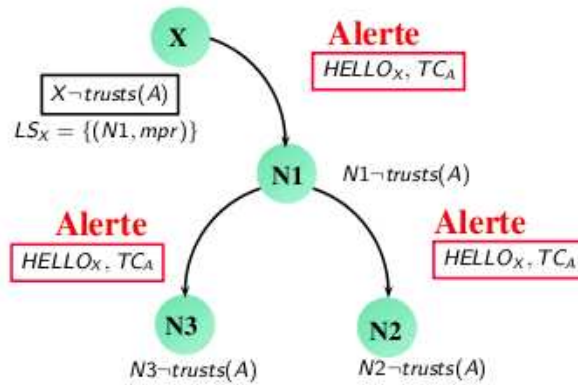


FIG. 5.10 – Distribution de la preuve de méfiance (alerte)

Chaque nœud recevant l'alerte comme une preuve de méfiance, va appliquer le même raisonnement pour détecter le nœud malveillant, comparer les informations fournies dans l'alerte avec la vision locale avant de la retransmettre. La figure 5.10 représente un exemple de distribution de la preuve de méfiance.

Par ailleurs, avant de retransmettre l'alerte, les informations qu'elle fournit doivent être corrélées avec la vision locale du nœud, afin de détecter les faux témoignages et refuser de retransmettre les alertes générées par les nœuds malveillants. Par exemple (figure 5.10), si le nœud x a déjà été détecté comme malveillant par $N1$, alors ce dernier ne doit pas retransmettre l'alerte.

Après la détection d'attaque, l'algorithme suivant résume l'ensemble des contremesures qui doivent être exécutées :

Algorithm 1 Contremesure et distribution de la preuve de confiance

Soit x un nœud raisonnant sur la confiance qui a détecté une attaque en comparant les messages M_A et M_B des deux nœuds A et B , et a déduit que A est un nœud malveillant : $x\text{-trusts}(IdP_A)$.

Si le nœud B a déjà été détecté comme malveillant : $IdP_B \in MN_x$, alors l'alerte est ignorée.

Sinon :

1. Si l'attaquant est un voisin symétrique alors :
 - A- Suppression du lien symétrique avec l'attaquant :
$$x\text{-trusts}(IdP_A) \Rightarrow LS_x = LS_x - \{(@IP_A, sym), (@IP_A, mpr)\}, NS_x = NS_x - \{IdP_A\}, MPRSS = MPRSS - \{IdP_A\}$$
Cela implique aussi la modification de l'ensemble des voisins à deux sauts $2HNS_x$ en supprimant les entrées de l'attaquant.
 - B- Enregistrement de l'identité de l'attaquant (IdP_A) :
$$MN_x = MN_x \cup \{IdP_A\}$$
 - C- Rejet des messages en provenance de l'attaquant : $HELLO_A$ et TC_A .
 - D- Si l'attaquant est MPR de x , alors x doit refaire sa sélection MPR en tenant compte des dernières modifications.
 2. x retransmet les messages M_A et M_B .
-

Notons, la distribution de la preuve de confiance ne permet pas la détection de l'attaque aux seuls nœuds qui ne l'ont pas détectée, mais elle permet aussi aux nœuds ayant établi une détection partielle de l'attaque, de déduire le nœud malveillant exactement, et arriver à une détection précise

5.2.2.1 Analyse et conséquences

Ces contremesures impliquent la retransmission des messages HELLO dans le cas d'une alerte, et par conséquent, à la réception de ces messages un lien asymétrique sera créé avec la source du message HELLO inclu dans l'alerte. Cette conséquence indirecte n'affecte pas les opérations de routage, puisque les liens asymétriques ne sont pas pris en considération pour la sélection MPR et le calcul de la table de routage, et sont supprimés après un certain moment si le lien symétrique n'est pas créé.

D'autre part, ces contremesures impliquent la sauvegarde des messages reçus pendant une durée déterminée, dans le but de pouvoir les comparer avec les messages reçus par la suite. Le processus de sauvegarde s'applique sur chaque message reçu, et n'enregistre que les informations nécessaires. Par exemple, le type de message (HELLO, TC ou DATA), la fraîcheur du message (selon l'horodatage dans $M\text{-SOLSR}$), le numéro de séquence, l'adresse source, et la liste des nœuds déclarés dans le message ainsi que leur type de lien lorsqu'il s'agit des messages HELLO. Si le message est déjà mémorisé avec un numéro de séquence inférieur ou égal, alors son contenu et sa durée de validité sont remis à jour. Si le message est reçu pour la première fois alors il est mémorisé avec une

durée de validité. Lorsque la durée de validité s'est écoulée le message est supprimé.

5.3 Conclusion

Nous avons proposé deux solutions complémentaires, la prévention pour pallier à certaines vulnérabilités du protocole OLSR, et des contremesures pour contrer et isoler les nœuds malveillants du réseau. Ces propositions répondent à nos besoins et correspondent au raisonnement sur la confiance qui a été fait par chaque nœud.

La prévention permet de valider le voisinage avant d'établir les relations de confiance de base, qui sont les relations établies avec les voisins symétriques. Ces relations sont importantes car elles sont la base de la sélection MPR et du routage. Cette étape permet ainsi à chaque nœud de constituer un voisinage à deux sauts de confiance et de détecter facilement les attaques concernant les liens de ce voisinage.

Lorsqu'une attaque est détectée, les contremesures permettent d'isoler le nœud malveillant du réseau et de garantir sa non participation aux opérations de routage. Pour permettre une détection globale, les contremesures permettent également à tous les nœuds qui détectent l'attaque précisément de partager leur informations en diffusant une preuve fiable constituée des informations relatives à la détection locale de l'attaque.

Notons que ces contremesures devant pouvoir être configurées en fonction du contexte de l'utilisation afin de préciser par exemple la durée de suppression du lien symétrique avec un nœud malveillant, la durée de validité des messages mémorisés et la gestion de diffusion des alertes (une seule fois, ou bien en boucle). Ces contremesures peuvent être configurées selon plusieurs critères : le taux de mobilité, la capacité mémoire des nœuds, et les besoins de routage...etc.

Dans la prochaine étape, nous allons présenter les résultats de simulation, et ainsi prouver l'efficacité du raisonnement basé sur la confiance pour la détection d'attaque, et l'efficacité des contremesures pour stopper l'attaque.

Chapitre 6

Validation par simulations du raisonnement basé sur la confiance

Dans les chapitres précédents, nous avons illustré l'efficacité du raisonnement basé sur la confiance pour la détection d'attaques en utilisant des simples scénarios, où la position de l'attaquant était importante pour la réussite de l'attaque. Cependant, une simulation avec des réseaux à grande échelle est nécessaire pour prouver la capacité et l'efficacité de l'approche, quelle que soit la position de l'attaquant et le nombre de nœuds dans le réseau.

Ce chapitre a pour objectif de démontrer l'efficacité du raisonnement basé sur la confiance pour sécuriser OLSR. Tout en préservant la spécification du protocole, nous proposons d'introduire dans chaque nœud OLSR l'ensemble des règles de confiance en vue d'évaluer le comportement des autres nœuds, et le cas échéant, de détecter les anomalies de comportement.

Nous commençons par introduction de l'outil de simulation. Ensuite, les étapes d'implémentation des règles de confiance, les contremesures et la gestion d'attaques dans les simulations (type et implémentations) seront présentés. Des discutons des résultats de simulations, et la capacité d'identifier les nœuds malveillants (attaquants) seront présentés a la fin du chapitre. .

6.1 Environnement de Simulation

Il existe plusieurs simulateurs de réseaux : Glomosim, NS2, Omnet++, Opnet Modeler...etc [sim]. Afin de choisir le simulateur le plus adapté à un réseau ad hoc (OLSR), plusieurs critères peuvent être considérés, en particulier :

- Précision des modèles
- Performance du moteur de simulation
- Passage à l'échelle
- Facilité d'utilisation (prise en main, description des scénarios, automatisation)
- Facilité de l'analyse des résultats
- Plate-forme d'exécution

- Type de licence

Les simulateurs NS-2 et Glomosim sont très sollicités et les plus populaires dans le domaine de la simulation des réseaux. NS-2 (Network Simulator) [NS2] est un simulateur à événements discrets. Il est développé en C++ avec une interface textuelle utilisant le langage OTcl (Object Tool Command Language) qui est une extension objet du langage de commande TCL (Tool Command Language).

GloMoSim est développé en langage PARSEC (Parallel Simulation Environnement for Complex Systems). PARSEC est un langage de simulation basé sur le langage C, développé par le laboratoire d'informatique parallèle à l'UCLA (University of California, Los Angeles) pour l'exécution séquentielle et parallèle des modèles de simulation à événement discret. Il peut être employé également comme langage de programmation parallèle. GloMoSim est basé sur un ensemble de modules et de bibliothèque. Chacune d'elles simule un aspect particulier des réseaux et une couche particulière du modèle OSI pour faciliter l'intégration des modèles développés aux différentes couches par différents développeurs.

Notre choix s'est porté sur GlomoSim (Global Mobile Simulator) [glo, BTA⁺, ZBG] pour sa flexibilité et sa configuration facile. De plus, c'est un environnement de simulation à grande échelle pour les réseaux sans fil avec une licence gratuite pour les institutions académiques.

Glomosim est livré avec un outil de visualisation appelé VT. Cet outil est codé en Java et s'interface directement à Glomosim. Il permet à l'utilisateur de visualiser les expériences et analyser le comportement des réseaux mobiles. L'outil est indépendant du simulateur, donc pas besoin de le modifier pour travailler avec le simulateur. Par ailleurs, le simulateur et l'outil de visualisation peuvent être modifiés facilement, ils disposent de plusieurs fonctionnalités et scénarios pour les réseaux sans fil et permettent de modifier les paramètres qui affectent ces fonctionnalités. Les détails d'installation et de configuration de GloMoSim sont présentés en annexe B.

6.1.1 Implémentation des règles de confiance et des contremesures

GloMoSim est livré avec l'implémentation de différents protocoles de routage sans l'OLSR. Cependant, une extension implémentant OLSR pour GloMoSim a été développée par l'université de Niigata [nii], cette extension peut être insérée facilement dans GloMoSim.

Pour intégrer le raisonnement sur la confiance, nous avons implémenté chaque règle dans le code, et quand une incohérence est détectée, le nœud enregistre les informations de l'alerte. A la fin de la simulation, chaque nœud affiche la table de routage correspondante, les informations des alertes émises et les incohérences détectées. Il est important de noter que les règles de confiance ne sont pas vérifiées dans les étapes d'initialisation, car au début des échanges il existe des incohérences qui sont légitimes et ne doivent pas être prises en compte. D'un autre côté, GloMoSim n'offre pas la possibilité de mettre en place l'identité prouvable, alors nous supposons dans la suite de ce chapitre que chaque nœud dispose d'une identité unique qui ne peut pas être usurpée.

Pour les contremesures, nous avons implémenté la première solution proposée dans le

chapitre 5. Cette solution repose sur la suppression des liens avec les nœuds malveillants afin de les isoler et de stopper leurs attaques.

6.1.2 Implémentation des attaques

Pour tester le fonctionnement des règles de confiance, nous avons implémenté différents scénarios d'attaques, et ajouté des paramètres relatifs au scénario d'attaque dans le fichier de configuration. Nous avons mis en place les quatre attaques suivantes :

1. L'attaquant déclare de faux liens pour être sélectionné comme MPR par le nœud cible et pouvoir contrôler ses messages.
2. L'attaquant ne déclare pas qu'il a été sélectionné comme MPR par d'autres nœuds.
3. L'attaquant sélectionné comme MPR ne va pas relayer les paquets des nœuds cibles.
4. L'attaque sur la table de routage (Chapitre 4).

Chaque attaque porte un identifiant unique entre 1 et 4, ce dernier est utilisé dans le fichier de configuration pour définir le scénario d'attaque appliqué dans la simulation. D'autres paramètres du scénario d'attaque sont présentés dans le tableau suivant :

Paramètre	Description
<i>ATTACKER_NUMBER</i>	Nombre de nœuds qui vont jouer le rôle d'attaquant dans le scénario, sa valeur par défaut est 1.
<i>ATTACKER</i> < <i>x</i> >	Identifiant de l'attaquant numéro <i>x</i> , tel que <i>x</i> est inférieur ou égal à <i>ATTACKER_NUMBER</i> . Si un attaquant n'est pas spécifié par l'utilisateur, alors il sera choisi aléatoirement au début de la simulation.
<i>ATTACK_DEFAULT</i>	Scénario d'attaque par défaut.
<i>ATTACK</i> < <i>x</i> >	Scénario d'attaque de l'attaquant numéro <i>x</i> , s'il n'est pas spécifié par l'utilisateur, alors il lui sera associé le scénario par défaut (<i>ATTACK_DEFAULT</i>).
<i>TARGET_NUMBER</i> < <i>x</i> >	Nombre des nœuds cibles de l'attaquant <i>x</i> .
<i>TARGET</i> < <i>x</i> > < <i>y</i> >	Nœud cible numéro <i>y</i> de l'attaquant <i>x</i> , tel que <i>y</i> est inférieur ou égal à <i>TARGET_NUMBER</i> < <i>x</i> >. si une cible n'est pas spécifiée par l'utilisateur, alors elle sera choisie aléatoirement au début de l'attaque.

Dans les simulations effectuées, les réseaux ad-hoc sont composés de plusieurs nœuds placés aléatoirement. Les attaquants sont sélectionnés aléatoirement, et chaque attaquant choisit aléatoirement un ensemble de nœuds cibles selon le scénario d'attaque retenu. Cependant, dans la mesure où le réseau doit être stable pour permettre à l'attaquant d'effectuer les scénarios d'attaques précédents, nous avons considéré les nœuds comme immobiles.

Dans un premier temps, nous allons discuter les résultats de simulations pour 50 nœuds, en appliquant le scénario d'attaque numéro 1 qui s'effectue selon les étapes suivantes :

1. L'attaquant A identifie la cible T , et ses voisins à 1 et 2 sauts.
2. L'attaquant A détecte ses voisins communs avec la cible T , et modifie son message HELLO en déclarant leurs voisins (des voisins commun entre lui et la cible) comme voisins symétriques en plus d'un nœud additionnel.

$$X : \forall Y \in NS_A \cap NS_T : HELLO = LS_A \cup NS_Y \cup \{X\}.$$

3. L'attaquant annonce les voisins à deux sauts de la cible dans son message TC (comme étant des nœuds qui l'ont sélectionnés comme MPR) :

$$\forall Y \in NS_A \cap NS_T, \forall Z \in NS_Y, Z \notin NS_A : TC_A = TC_A \cup Z, X.$$

Selon la spécification OLSR, la cible doit sélectionner l'attaquant comme MPR parce qu'il fournit l'accès au nœuds Z et X , et donc l'attaquant peut contrôler certains messages de la cible.

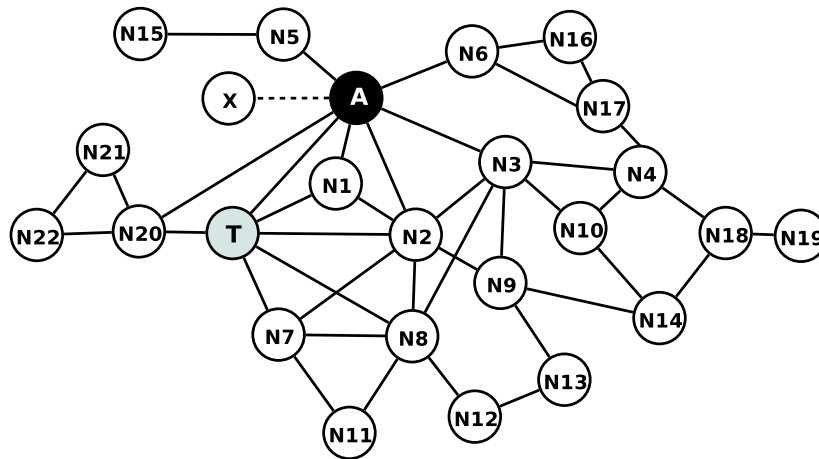


FIG. 6.1 – Exemple de scénario : A est l'attaquant, et T est la cible

Afin d'illustrer ce scénario d'attaque, nous allons prendre l'exemple présenté dans la figure 6.1. L'attaque prend place en suivant les étapes suivantes :

1. L'attaquant A identifie la cible T , ses voisins $\{N1, N2, N7, N8, N20\}$ et ses voisins à 2 sauts $\{N3, N7, N8, N9, N12, N11, N21, N22\}$.
2. L'attaquant A détecte les voisins qu'il a en commun avec la cible T : $(N1, N2, N20)$, et modifie son message HELLO pour annoncer leurs voisins $(N7, N8, N9, N21, N22)$ comme ses voisins symétriques : $NS_A = \{T, N1, N2, N3, N5, N6, N20, N7, N8, N9, N21, N22, X\}$ (X est un nœud fictif).
3. L'attaquant déclare les voisins à 2 sauts de la cible dans ses messages TC (comme étant des nœuds qui l'ont sélectionnés comme MPR) : $N7, N8, N9, N21, N22 \in TC_A$, figure 6.2.

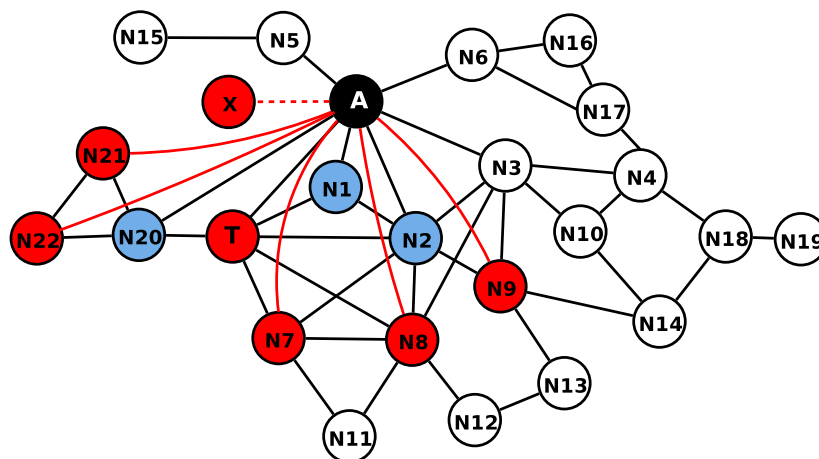


FIG. 6.2 – Etablissement de l'attaque

Selon la spécification OLSR, la cible T sera obligée de choisir l'attaquant A comme MPR, autorisant ainsi à l'attaquant le contrôle de certains de ses flux de données. Dans cet exemple, les nœuds touchés par l'attaque sont la cible T , les voisins à deux sauts de la cible $\{N7, N8, N9, N21, N22\}$ puisqu'ils sont annoncés comme voisins symétriques de l'attaquant, et les voisins de ses nœuds car ils sont affectés indirectement par l'attaque (ils verront que certains nœuds sont annoncés comme voisins de l'attaquant alors qu'ils ne le sont pas).

6.2 Résultats de simulation

Dans toutes les simulations suivantes, les attaquants et les cibles sont choisis aléatoirement, en tenant compte que les cibles doivent être choisies parmi les voisins de l'attaquant (selon le scénario d'attaque numéro 1).

Les résultats de simulation sont présentés sous la forme de taux de détection, en comparant le nombre des nœuds pouvant détecter l'attaque (précisément ou partiellement) avec le nombre total des nœuds du réseau.

Le but de ces simulations est de montrer qu'un attaquant peut avoir une ou plusieurs cibles, et que ces cibles sont capables de le détecter en raisonnant sur la confiance dans le comportement des autres nœuds. Dans le cas de plusieurs attaquants, ces résultats prouvent que les nœuds, raisonnant sur la confiance, sont capables de différencier les comportements anormaux et détecter les attaquants séparément.

L'ensemble des nœuds concernés dépend du scénario d'attaque et du raisonnement sur la confiance défini dans les chapitres précédents, car chaque scénario touche et/ou implique un ensemble particulier de nœuds. En appliquant le premier scénario d'attaque, les nœuds concernés qui doivent détecter l'attaque utilisant le raisonnement sur la confiance sont les suivants :

1. Le nœud cible, qui est le premier et le plus concerné car il est la cible de l'attaque,

2. Les voisins à 2 sauts de la cible, qui sont annoncés comme voisins symétriques par l'attaquant et aussi comme des nœuds qui ont sélectionné l'attaquant comme MPR alors qu'ils ne le sont pas,
3. Les voisins des nœuds déclarés faussement comme voisins de l'attaquant, en comparant l'information locale avec le message TC de l'attaquant,
4. Les nœuds qui peuvent comparer le voisinage de l'attaquant et celui des voisins de la cible (ceux qui fournissent l'accessibilité aux voisins à deux sauts de la cible et qui sont annoncés par l'attaquant).

6.2.0.1 Etape de la détection

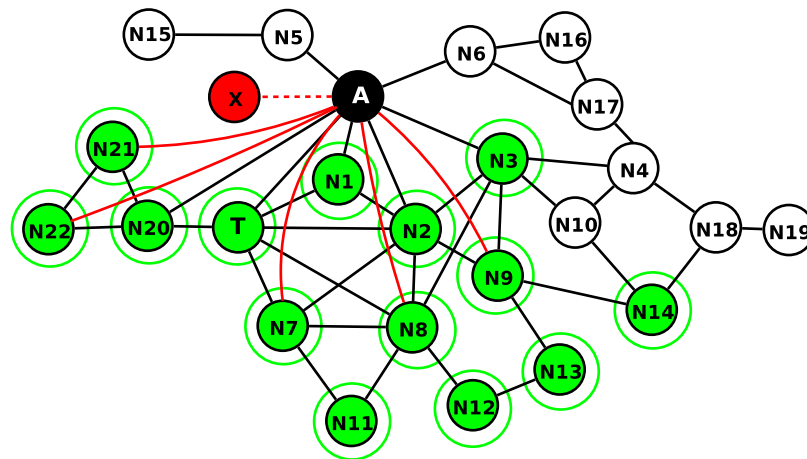


FIG. 6.3 – Détection de l'attaque

Par exemple, nous prenons le réseau présenté dans la figure (6.3), où l'attaquant est le nœud A , et la cible est le nœud T . En utilisant le raisonnement sur la confiance, les nœuds concernés sont les suivants :

1. La cible : le nœud T détecte l'attaque en utilisant les formules suivantes :
 - La formule 4.8 : le nœud détecte une incohérence entre les messages HELLO de $N7, N8$ et A , où $N7, N8 \in NS_A$ alors que $A \notin NS_{N7}, A \notin NS_{N8}$.
 - La formule 4.6 : le nœud détecte une incohérence entre les messages HELLO de $N7, N8$ et le message TC de A , où $N7, N8 \in TC_A$ alors que $A \notin NS_{N7}$ et $A \notin NS_{N8}$.
 - La formule 4.15 : dans cet exemple, le nœud $N9$ va sélectionner $N2$ comme MPR et les nœuds $N21, N22$ vont sélectionner $N20$ comme MPR. A la réception des messages TC de $N2$ et $N20$, La cible va détecter une incohérence (4.15), parce que le voisinage de $N2$ et de $N20$ sont inclus dans celui de l'attaquant, et donc les nœuds $N2$ et $N20$ ne devraient pas être choisis comme MPR (c'est plutôt l'attaquant qui devrait l'être puisqu'il déclare être voisin de $N9, N21, N22$).

2. Les voisins à deux sauts de la cible annoncés comme voisins symétriques et comme étant des nœuds qui ont sélectionné l'attaquant comme MPR : les nœuds ($N7, N8, N9, N21, N22 \in TC_A$) détectent aussi l'attaque. Par exemple, le nœud $N7$ détecte l'attaquant en utilisant les formules 4.8 et 4.6 : quand il reçoit le message TC de l'attaquant, il verra qu'il a été déclaré dedans et va détecter une incohérence parce que l'attaquant n'est pas un voisin symétrique et il ne l'a pas sélectionné comme MPR : $A \notin MPRSS_{N7}$.
3. Les voisins des faux voisins déclarés par l'attaquant détectent l'attaque : Par exemple, le nœud $N2$ détecte une incohérence en utilisant la formule 4.6 à la réception du message TC de l'attaquant parce que $N7, N8, N9 \in TC_A$ alors que $A \notin NS_{N7}, A \notin NS_{N8}, A \notin NS_{N9}$.
4. Les nœuds qui peuvent comparer le voisinage de l'attaquant et celui des voisins de la cible : Par exemple, quand le nœud $N1$ reçoit les messages TC de l'attaquant et de $N2$, il détecte une incohérence selon la formule 4.15 car le voisinage de $N2$ est inclus dans celui de l'attaquant.

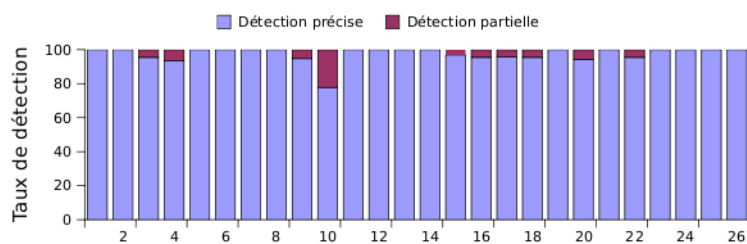


FIG. 6.4 – Taux de détection : 50 nœuds, un attaquant et une cible

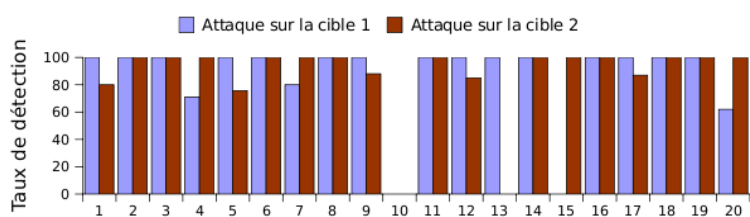


FIG. 6.5 – Taux de détection : 50 nœuds, un attaquant et deux cibles

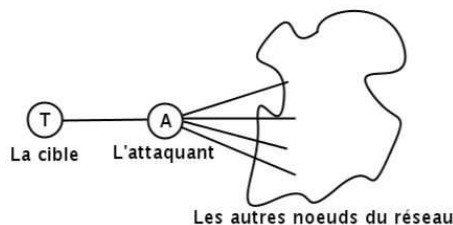


FIG. 6.7 – Exemple où la cible est isolée du réseau par l'attaquant

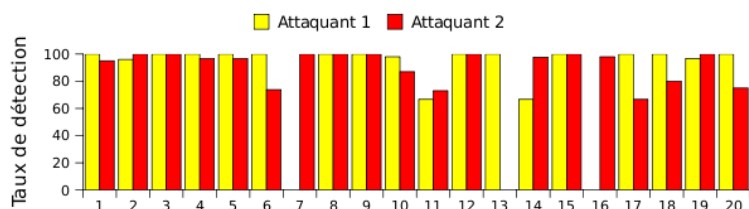


FIG. 6.6 – Taux de détection : 50 nœuds, deux attaquants et une cible chacun

Les résultats de détection sont présentés dans les figures 6.4, 6.5 et 6.6. Dans le cas d'un seul attaquant et une seule cible, le taux de détection est généralement 100%. Toutefois, il existe un seul cas où le taux de détection est de 0%, il s'agit du cas où la cible est complètement isolée par rapport aux autres nœuds du réseau et/ou quand l'attaquant est toujours MPR unique de la cible (même avant d'appliquer l'attaque). Il est évident que dans cette situation l'objectif d'une telle attaque n'a aucun sens, car l'attaquant est déjà MPR unique de la cible. La réalisation de l'attaque ne modifie pas la topologie du réseau, et donc elle n'est pas détectée. Le seul point sur lequel l'attaquant a menti, est son lien avec le nœud fictif (qui n'a aucune utilité dans cette situation). Ce mensonge ne peut pas être détecté, car selon la spécification de OLSR, il est possible qu'un nœud soit un voisin/mpr unique d'un autre nœud. Ainsi, un nœud malveillant peut créer autant de voisins fictifs qu'il veut, puisqu'il est le seul voisin symétrique du nœud fictif, et les autres nœuds du réseau n'ont aucun moyen de vérifier l'existence physique de ce dernier.

Par exemple, dans la figure 6.7, l'attaquant sera toujours MPR unique de la cible, et cette dernière n'a aucun moyen de détecter l'attaque car elle est isolée du reste du réseau et ne reçoit que les informations fournies par l'attaquant. En effet, le raisonnement sur la confiance repose sur la comparaison de plusieurs visions du réseau, et dans cette situation, la cible ne peut comparer sa vision qu'avec celle de l'attaquant, or ces dernières restent cohérentes.

Cette situation représente une attaque d'écoute passive, car elle permet à l'attaquant d'espionner le flux entrant et sortant de la cible, et elle ne sera détectée que lorsque l'attaquant devient actif. Par exemple, en refusant de retransmettre les paquets.

Dans le cas de plusieurs cibles, les résultats de détection concernent l'attaque contre chaque cible séparément. En effet, un nœud est concerné par l'attaque contre $T1$ ou $T2$

seulement lorsqu'il est impliqué dans le scénario d'attaque contre l'un des deux. Ainsi, un nœud concerné doit détecter chaque attaque séparément.

Il est important de noter que dans certaines simulations le taux de détection n'atteint pas les 100% pour plusieurs raisons :

- Dans certains scénarios, le temps de simulation ne suffit pas à tout les nœuds pour détecter l'attaque,
- L'environnement choisit l'attaquant aléatoirement. Ainsi, l'établissement de l'attaque peut échouer si l'attaquant choisi n'a pas de voisins, et la simulation peut se terminer sans que l'attaque ait démarré.

Par ailleurs, certains nœuds n'arrivent pas à détecter l'attaquant précisément (détection partielle). Cette situation peut être résolue par un système de partage d'informations sur la confiance (chapitre 5, section 5.2.2, page 72). Ainsi, les nœuds qui détectent précisément peuvent distribuer une preuve d'attaque et permettre aux autres nœuds de détecter l'attaque également.

6.2.1 Etape des contremesures

Après la détection de l'attaque, nous avons implémenté la contremesure décrite dans le chapitre 5, (section 5.2.1, page 71), afin de permettre aux nœuds d'isoler un voisin si son comportement est malveillant.

Dans cette partie, nous démontrons que lorsque les nœuds détectent l'attaque et appliquent cette contremesure, alors l'attaquant est isolé et l'attaque est stoppée et ne se répercute pas sur les autres nœuds du réseau. Puisque cette contremesure n'est appliquée que par les voisins de l'attaquant, nous nous intéressons dans la suite à la vision de l'attaquant et de ses voisins seulement.

Prenons l'exemple présenté dans la figure 6.1 (page 80), la vision des voisins de l'attaquant avant l'attaque est la suivante :

Le nœud A , $NS_A = \{T, N1, N2, N3, N5, N6, N20\}$	$MPRS_A = \{N20, T, N3, N5, N6\}$
Le nœud T , $NS_T = \{A, N20, N1, N2, N7, N8\}$	$MPRS_T = \{N20, N2, N8\}$
Le nœud $N1$, $NS_{N1} = \{A, T, N2\}$	$MPRS_{N1} = \{A, N2\}$
Le nœud $N2$, $NS_{N2} = \{A, T, N1, N3, N7, N8, N9\}$	$MPRS_{N2} = \{A, N8, N3, N9\}$
Le nœud $N3$, $NS_{N3} = \{A, N2, N8, N9, N10, N4\}$	$MPRS_{N3} = \{A, N8, N9, N4\}$
Le nœud $N5$, $NS_{N5} = \{A, N15\}$	$MPRS_{N5} = \{A\}$
Le nœud $N6$, $NS_{N6} = \{A, N16, N17\}$	$MPRS_{N6} = \{A, N17\}$
Le nœud $N20$, $NS_{N20} = \{A, T, N21, N22\}$	$MPRS_{N20} = \{A, T\}$

Lorsque l'attaque est établi (figure 6.2, page 81), la cible choisit l'attaquant comme MPR unique, et la vision de l'attaquant et ses voisins prend la forme suivante :

Le nœud A , $NS_A = \{T, N1, N2, N3, N5, N6, N20, N21, N22, N7, N8, N9, x\}$	$MPRS_A = \{N5, N6\}$
Le nœud T , $NS_T = \{A, N20, N1, N2, N7, N8\}$	$MPRS_T = \{\mathbf{A}\}$
Le nœud $N1$, $NS_{N1} = \{A, T, N2\}$	$MPRS_{N1} = \{A\}$
Le nœud $N2$, $NS_{N2} = \{A, T, N1, N3, N7, N8, N9\}$	$MPRS_{N2} = \{A, N8, N3, N9\}$
Le nœud $N3$, $NS_{N3} = \{A, N2, N8, N9, N10, N4\}$	$MPRS_{N3} = \{A, N8, N9, N4\}$
Le nœud $N5$, $NS_{N5} = \{A, N15\}$	$MPRS_{N5} = \{A\}$
Le nœud $N6$, $NS_{N6} = \{A, N16, N17\}$	$MPRS_{N6} = \{A, N17\}$
Le nœud $N20$, $NS_{N20} = \{A, T, N21, N22\}$	$MPRS_{N20} = \{A\}$

En établissant le raisonnement sur la confiance dans ce scénario, seuls les nœuds voisins $\{T, N20, N1, N2, N3\}$ vont détecter l'attaque, car ils sont les seuls concernés par ce scénario d'attaque. En appliquant les contremesures, ces voisins vont supprimer leur lien symétrique avant l'attaquant et refuser les messages de ce dernier. Ainsi, la vision devient la suivante :

Le nœud A , $NS_A = \{N5, N6, x\}$	$MPRS_A = \{N5, N6\}$
Le nœud T , $NS_T = \{N20, N1, N2, N7, N8\}$	$MPRS_T = \{N20, N2, N8\}$
Le nœud $N1$, $NS_{N1} = \{T, N2\}$	$MPRS_{N1} = \{T, N2\}$
Le nœud $N2$, $NS_{N2} = \{T, N1, N3, N7, N8, N9\}$	$MPRS_{N2} = \{T, N8, N3, N9\}$
Le nœud $N3$, $NS_{N3} = \{N2, N8, N9, N10, N4\}$	$MPRS_{N3} = \{N2, N8, N9, N4\}$
Le nœud $N5$, $NS_{N5} = \{A, N15\}$	$MPRS_{N5} = \{A\}$
Le nœud $N6$, $NS_{N6} = \{A, N16, N17\}$	$MPRS_{N6} = \{A, N17\}$
Le nœud $N20$, $NS_{N20} = \{T, N21, N22\}$	$MPRS_{N20} = \{T\}$

Par conséquent, la topologie du réseau prend la forme présentée dans la figure (6.8).

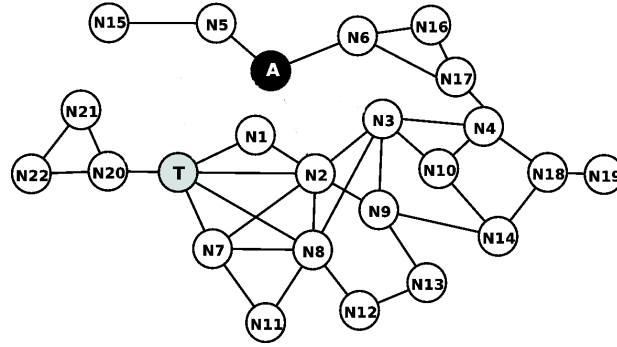


FIG. 6.8 – Vision de la topologie après la détection de l'attaque

Les nœuds ayant détecté l'attaque ont isolé le nœud malveillant en supprimant les liens symétriques. Ainsi, l'attaquant ne pourra appliquer aucune action malveillante concernant la cible T .

Il est important de noter que les nœuds $N5$ et $N6$ n'ont pas détecté cette attaque car il ne sont pas affectés dans ce scénario (leur sélection MPR avant et après l'attaque est identique). Mais lorsqu'un mécanisme de distribution de preuve de confiance est établi, ils vont pouvoir détecter l'attaque et appliquer la contremesure pour isoler le nœud malveillant.

A l'heure actuelle, nous n'avons pas pu implémenter sous GloMoSim la distribution de preuve de voisinage et la preuve de confiance, car le simulateur ne permet pas à un nœud de retransmettre les messages HELLO reçus des autres nœuds. Par ailleurs, si ce mécanisme est mise en place, le nœud $N6$ (et les autres nœuds du réseau) va recevoir une preuve (par exemple : $HELLO_{N9} + TC_A$), et déduire que le nœud A est malveillant.

Par contre, il est possible que les nœuds $N5$ et $N15$ ne puissent pas détecter l'attaque, car leur seul lien avec le reste du réseau est l'attaquant. Ce dernier peut intercepter la preuve de confiance et ne pas la retransmettre. Ainsi, l'attaque ne pourra pas

être détectée par les nœuds $N5$, et $N15$.

6.2.2 Analyse du taux de faux positifs

Chaque règle de confiance nécessite une vérification, nous appelons un faux positif le cas où une vérification révèle une incohérence pendant l'absence d'une attaque. A l'initialisation du réseau et au début des échanges, beaucoup d'incohérences peuvent être révélées car chaque nœud est en train de construire sa vision du réseau, et la partie qui est contruite dans un nœud n'est pas forcément cohérente avec celle d'un autre nœud. Par exemple, lorsqu'un nœud x sélectionne son voisin y comme MPR, ce dernier ne va pas le sélectionner comme sélecteur MPR tant qu'il n'aura pas reçu le message HELLO de x . Ainsi, entre la sélection MPR du nœud x et la réception du message HELLO par y indiquant les nœuds MPR sélectionnés, la vision du nœud x est incohérente avec celle de son MPR y . Cette situation se produit au début des échanges et produit des faux positifs mais ne représente pas une attaque. Dans les simulations précédentes, nous avons testé des scénarios sans mobilité pour pouvoir appliquer l'attaque, et nous avons mis en place un minuteur (*timer*), qui déclenche le raisonnement sur la confiance après une durée du début de la simulation, afin d'éviter les faux positifs et d'attendre que le réseau soit stabilisé, c'est à dire quand la découverte de voisinage, la sélection MPR, et le signalement des sélecteur MPR de chaque nœud sont faits.

Dans le cas d'un réseau non-mobile, le raisonnement sur la confiance commence après l'activation d'un minuteur d'une durée déterminée pour éviter les faux positifs déclenchés au début des échanges. Cependant, dans le cas de mobilité, les faux positifs sont détectés à chaque fois que le nœud se déplace et initialise ses liens. Glomosim offre la possibilité de régler la vitesse de déplacement des nœuds, ceci en définissant un interval de vitesses minimum et maximum et la durée de pause après la fin d'un déplacement.

Pour évaluer le taux de faux positifs, nous avons effectué des simulations pour des nœuds mobiles avec différentes vitesses et différentes pauses de déplacement. Ensuite, nous avons calculé le taux de faux positifs détecté par chaque nœud, et le taux final pour chaque simulation est égale à la moyenne des taux calculés par tous les nœuds. Les résultats sont représentés comme suite dans le tableau :

Vitesse	Pause de déplacement (MOBILITY-WP-PAUSE)										
	0s	5s	10s	15s	20s	25s	30s	35s	40s	45s	50s
0 - 1	0,74%	0,75%	0,79%	0,78%	0,80%	0,77%	0,77%	0,75%	0,66%	0,66%	0,78%
2 - 3	1,76%	1,84%	1,85%	1,81%	1,94%	1,88%	1,84%	1,74%	1,76%	1,72%	1,77%
4 - 5	2,86%	2,83%	2,85%	2,88%	2,75%	2,79%	2,65%	2,66%	2,63%	2,47%	2,55%
6 - 7	3,94%	3,71%	3,64%	3,56%	3,45%	3,40%	3,31%	3,36%	3,30%	3,27%	3,38%
8 - 9	4,43%	4,18%	4,12%	4,25%	3,98%	4,16%	4,01%	4,15%	3,94%	3,91%	3,90%
10 - 11	4,92%	4,72%	4,46%	4,70%	4,61%	4,48%	4,56%	4,41%	4,26%	4,33%	4,29%
12 - 13	5,47%	5,22%	5,01%	5,25%	5,10%	5,19%	4,86%	4,76%	4,56%	4,29%	4,49%
14 - 15	5,90%	5,69%	5,57%	5,51%	5,48%	5,28%	5,14%	4,91%	4,71%	4,67%	4,63%
16 - 17	6,24%	6,08%	5,97%	5,79%	5,71%	5,31%	5,26%	5,19%	5,06%	4,66%	4,66%
18 - 19	6,58%	6,51%	6,28%	6,20%	5,84%	5,65%	5,42%	5,24%	5,03%	5,00%	4,66%
20 - 21	7,07%	6,59%	6,50%	6,33%	5,92%	5,69%	5,66%	5,32%	5,16%	5,18%	5,06%
22 - 23	7,38%	6,97%	6,72%	6,50%	6,14%	5,88%	5,69%	5,48%	5,41%	5,31%	5,21%
24 - 25	7,73%	7,27%	7,29%	6,62%	6,18%	6,19%	5,93%	5,77%	5,58%	5,54%	5,08%
26 - 27	7,92%	7,63%	7,33%	6,88%	6,29%	6,27%	6,01%	5,78%	5,65%	5,40%	5,21%
28 - 29	8,22%	7,94%	7,53%	6,94%	6,53%	6,24%	6,24%	5,83%	5,67%	5,38%	5,19%
30 - 31	8,39%	8,16%	7,83%	7,00%	6,62%	6,62%	6,22%	5,88%	5,79%	5,59%	5,32%
32 - 33	8,72%	8,30%	8,04%	7,24%	6,84%	6,69%	6,22%	6,04%	5,79%	5,69%	5,41%
34 - 35	8,72%	8,46%	8,00%	7,21%	6,97%	6,53%	6,22%	6,18%	5,92%	5,68%	5,49%
36 - 37	8,82%	8,63%	8,19%	7,51%	7,00%	6,54%	6,34%	6,22%	5,96%	5,69%	5,46%
38 - 40	9,02%	8,86%	8,23%	7,72%	7,36%	6,68%	6,65%	6,19%	6,01%	5,91%	5,56%

TAB. 6.1 – Taux de faux positifs dans un réseau à nœuds mobiles

Les diagrammes suivants représentent le taux de faux positifs par rapport à la vitesse des nœuds (figure 6.9), et par rapport à la durée de pause (figure 6.10) :

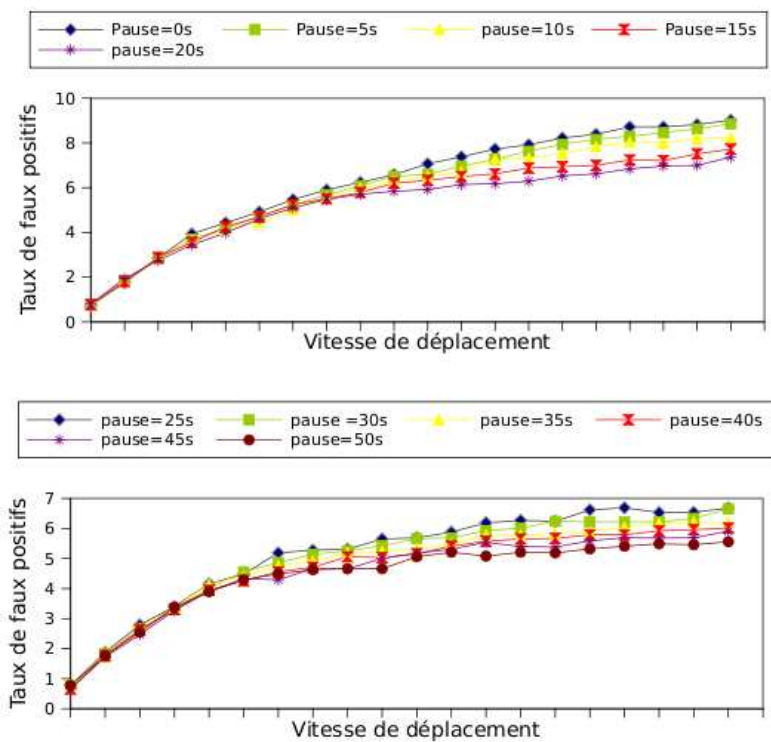


FIG. 6.9 – Diagramme représentant le taux de faux positifs selon différentes vitesses

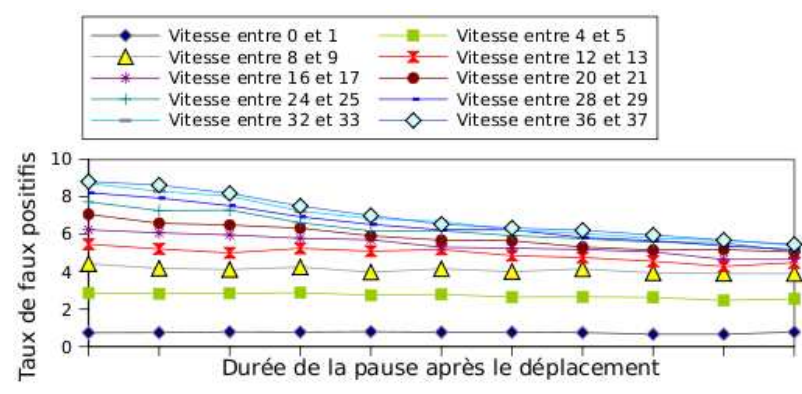


FIG. 6.10 – Taux de faux positifs par rapport à la pause de déplacement

Dans la spécification OLSR, les messages HELLO sont envoyés chaque 3 secondes, et un lien symétrique est créé en environ 6 secondes. Ainsi, si la vitesse d'un nœud est telle qu'il ne peut rester plus de 6 secondes dans la portée d'un autre nœud, alors le lien symétrique ne sera pas établi et il y aura une rupture de liens lorsque le nœud se

déplacera à nouveau. Ce qui explique l'augmentation du taux de faux positifs lorsque la mobilité est rapide (la figure 6.9).

Cette déduction est confirmée dans le deuxième diagramme (figure 6.10), lorsque la pause est petite, le nœud change rapidement de position (quelle que soit sa vitesse), ce qui implique la rupture fréquente des liens qui ont été créés pendant la pause et la détection de faux positifs par ses voisins. Par conséquent, la durée de la pause est inversement proportionnelle au taux de faux positifs.

En résumé, afin d'adapter le raisonnement sur la confiance avec la mobilité des nœuds, il est important de mettre en place un minuteur spécial. Ce dernier doit être réinitialisé à chaque nouveau déplacement. De plus, le nœud doit attendre une durée déterminée avant de déclencher une règle pour s'assurer que cette situation ne concerne pas un lien qui a été rompu. Par exemple, si le MPR sélectionné ne génère pas de message TC, le nœud doit d'abord vérifier que le lien symétrique avec ce MPR n'a pas été rompu.

6.3 Conclusion

En utilisant le simulateur GlomoSim, nous avons mis en place des nœuds qui sont en mesure de raisonner en utilisant nos règles de confiance pour valider la sélection des MPRs. La simulation basée sur les scénarios d'attaque montre l'efficacité de l'utilisation de la notion de méfiance pour détecter des attaques connues contre le protocole OLSR et cela en se basant sur des observations locales. Ce résultat a motivé l'extension de l'approche pour la validation de la table de routage OLSR des nœuds et la simulation de situations plus complexes avec un plus grand nombre de nœuds.

Avec des scénarios plus aléatoires en ce qui concerne le nombre de nœuds, le nombre d'attaquants et le choix des formes d'attaque, la simulation a démontré l'efficacité du raisonnement basé sur la confiance pour la détection d'attaque. Les résultats nous permettent de mettre en place des vérifications que chaque nœud peut effectuer pour évaluer le bon comportement des autres nœuds et de détecter les attaques contre le protocole OLSR. Il est important de mentionner que le protocole OLSR (messages) est inchangée, et notre approche reste compatible avec le protocole OLSR de base.

D'un autre côté, ces résultats prouvent également que les contremesures permettent d'isoler l'attaquant du réseau, et assurent le bon fonctionnement du routage après la détection d'attaque.

Chapitre 7

Conclusions

Dans ce mémoire, nous avons présenté les résultats de nos travaux portant sur la sécurité des réseaux ad-hoc. En particulier, nous avons choisi d'étudier le protocole OLSR, et nous nous sommes intéressés au concept de la gestion de confiance (*trust management*) comme une solution de sécurité. Ce concept s'adapte particulièrement à la nature mobile, ad-hoc, distribuée et auto-organisée des réseaux ad-hoc.

La première contribution a été l'analyse de la confiance implicite dans le protocole OLSR, et l'expression des relations de confiance avec un langage formel. Cette analyse a fait ressortir des possibles mesures pour rendre OLSR plus fiable et cela en exploitant des opérations et des informations déjà existantes dans le protocole.

Afin de mettre en place un mécanisme de méfiance envers les comportements suspects, nous avons développé dans la seconde contribution un raisonnement basé sur la confiance permettant la corrélation entre les informations fournies dans les messages reçus. L'intégration de ce raisonnement permet à chaque nœud de vérifier la cohérence du comportement des autres nœuds et de valider les relations de confiance établies implicitement.

La troisième contribution vient compléter la deuxième en proposant deux solutions complémentaires : la prévention contre certaines vulnérabilités du protocole, et les contremesures après la détection de comportement malveillant. Nous nous sommes intéressés particulièrement à la vulnérabilité due à l'absence de validation des liens et à la facilité d'usurper les identités dans OLSR. Pour cela, nous avons proposé un mécanisme qui repose sur le concept d'identité prouvable afin d'éviter l'usurpation d'identité dans OLSR, et un mécanisme de validation des liens afin de permettre à chaque nœud de prouver les liens qu'il déclare. Ces mécanismes ne sont pas suffisants pour pallier à toutes les vulnérabilités de OLSR, d'où le besoin de contremesures dans le cas d'une attaque. Pour cela, nous avons proposé d'isoler le nœud malveillant en rejetant ses messages et en distribuant une preuve de l'attaque dans le réseau afin de permettre à tous les autres nœuds d'identifier le nœud malveillant.

Ensuite, nous avons présenté les résultats de simulations afin de montrer l'efficacité du raisonnement basé sur la confiance et des contremesures pour stopper et isoler les

nœuds se comportant mal.

En résumé, nous avons démontré que le raisonnement basé sur la confiance permet de vérifier si le comportement des autres nœuds du réseau respecte la spécification d'OLSR. Ce résultat nous a permis de s'assurer du bon fonctionnement des opérations de routage et de garantir la validité de la topologie du réseau (la table de routage).

Perspectives

En perspective, nous pensons qu'il est intéressant d'établir plus d'exemple de simulation (par exemple, implémenter l'ensemble des contremesures proposées, étudier d'autres types d'attaques...etc.) et de vérifier le degré d'automatisation du raisonnement sur la confiance afin de mesurer l'impact de ces solutions sur le protocole, tout en préservant l'auto-organisation et la dynamicité de l'environnement ad-hoc.

Notre approche peut être plus généralement appliquée à d'autres protocoles de contrôle en environnement spontané et auto-organisé. La spécification explicite de relations de confiance permet de déterminer si les hypothèses nécessaires à l'opération d'un protocole sont réalistes et permet de comparer différents protocoles selon le critère des exigences de confiance qu'ils requièrent [YKB93]. Par ailleurs, les relations explicites de confiance peuvent être utilisées pour l'analyse formelle de la correction des vulnérabilités d'un protocole.

En ce qui concerne l'usage d'une spécification explicite de relations de confiance, il pourrait être intéressant de comparer OLSR à d'autres protocoles, par exemple AODV, sous l'optique de la confiance, et voir ce que ça peut nous rapporter comme informations supplémentaires sur l'aspect sécurité des deux protocoles.

Dans ce travail, nous avons défini plusieurs classes de confiance et seulement deux niveaux de confiance : confiance (*trusts*) ou méfiance (*-trusts*). En perspective, il pourrait être possible d'étudier les différents règles de confiance et de définir des niveaux de confiance et de méfiance selon l'action concernée. Par exemple, permettre à un nœud d'accorder sa confiance à un autre nœud pour effectuer une action précise même s'il ne lui fait pas confiance pour une autre action. D'autant plus que cette notion existe déjà dans OLSR : l'analyse de la confiance (chapitre 3) a démontré que la confiance accordée aux nœuds MPR est plus importante que celle accordée aux voisins symétriques.

Dans [CH07], les auteurs ont proposé une version multi-chemin d'OLSR. Il pourrait être intéressant d'appliquer notre approche pour étudier les vulnérabilités de cette version. Par contre, il est nécessaire de créer un nouveau raisonnement basé sur la confiance pour sécuriser le routage, car le raisonnement que nous avons proposé s'adapte à la version OLSR de base. D'un autre côté, la confiance pourrait être prise en compte dans la sélection des routes et des MPR.

Au final, il existe plusieurs aspects de la confiance qui n'ont pas été exploités dans cette thèse. Par exemple, le concept de la confiance dans les réseaux sociaux. Cette idée pourrait inspirer des application dans le domaine des réseaux ad hoc. Au lieu de vérifier le comportement individuel, un nœud pourrait surveiller le comportement d'un groupe de nœuds afin de décider s'il peut faire confiance ou pas à l'un ou plusieurs d'entre eux.

Annexe A

Raisonnement sur la confiance avec la notation de l'identité prouvable

Dans cette annexe, nous allons reformuler les règles du raisonnement sur la confiance (présentées dans le chapitre 4) en tenant compte du mécanisme d'identité prouvable. Pour ce faire, nous partons des définitions suivantes :

- $@IP_x$: signifie l'adresse IP du nœud x .
- IdP_x : signifie l'identité prouvable du nœud x .

La formule 4.1 (chapitre 4, page 41) devient :

$$\begin{aligned} & x \xleftarrow{HELLO_y} y, x \xleftarrow{TC_y} y, \exists z, @IP_z \in TC_y : \\ & (@IP_z, sym) \notin HELLO_y \text{ et } (@IP_z, mpr) \notin HELLO_y \\ & \quad \downarrow \\ & x \neg trusts(IdP_y) \end{aligned} \tag{A.1}$$

La formule 4.2 (chapitre 4, page 41) devient :

$$x \xleftarrow{TC_y} *, @IP_x \in TC_y, IdP_y \notin NS_x \Rightarrow x \neg trusts(IdP_y) \tag{A.2}$$

La formule 4.3 (chapitre 4, page 42) devient :

$$x \xleftarrow{TC_y} *, @IP_x \in TC_y, IdP_y \notin MPRS_x \Rightarrow x \neg trusts(IdP_y) \tag{A.3}$$

La formule 4.4 (chapitre 4, page 42) devient :

$$\begin{aligned} & \forall z, w \in MANET, x \xleftarrow{(TC_y)_z} *, x \xleftarrow{(TC_y)_w} *, (TC_y)_z \neq (TC_y)_w \\ & \quad \downarrow \\ & x \neg trusts(\{IdP_z, IdP_w\}) \end{aligned} \tag{A.4}$$

La formule 4.5 (chapitre 4, page 42) devient :

$$\begin{aligned} x \xrightarrow{HELLO_z} z, x \xleftarrow{TC_y} *, IdP_z \in NS_x, @IP_z \in TC_y, (@IP_y, sym) \notin HELLO_z \\ \Downarrow \\ x \neg trusts(IdP_y) \end{aligned} \quad (A.5)$$

La formule 4.6 (chapitre 4, page 42) devient :

$$\begin{aligned} x \xrightarrow{HELLO_z} z, x \xleftarrow{TC_y} *, IdP_z \in NS_x, @IP_z \in TC_y, (@IP_y, mpr) \notin HELLO_z \\ \Downarrow \\ x \neg trusts(IdP_y) \end{aligned} \quad (A.6)$$

La formule 4.8 (chapitre 4, page 43) devient :

$$\begin{aligned} x \xleftarrow{HELLO_y} y, x \xleftarrow{HELLO_z} z, \\ (@IP_z, sym) \in HELLO_y \text{ ou } (@IP_z, mpr) \in HELLO_y, \\ IdP_z \in NS_x, (@IP_y, sym) \notin HELLO_z, (@IP_y, mpr) \notin HELLO_z, \\ \Downarrow \\ x \neg trusts(IdP_y, IdP_z) \end{aligned} \quad (A.7)$$

La formule 4.11 (chapitre 4, page 44) devient :

$$\begin{aligned} IdP_y \in MPRS_x, \quad (x \xrightarrow{TC_y} y) \text{ ou } \Rightarrow x \neg trusts(IdP_y) \\ (x \xleftarrow{TC_y} y, @IP_x \notin TC_y) \end{aligned} \quad (A.8)$$

La formule 4.12 (chapitre 4, page 44) devient :

$$\begin{aligned} IdP_y \in MPRS_x, \quad (x \xrightarrow{TC_x} *, x \xrightarrow{(TC_x)_y} y) \text{ et } \Rightarrow x \neg trusts(IdP_y) \\ (x \xrightarrow{DATA_x} *, x \xrightarrow{(DATA_x)_y} y) \end{aligned} \quad (A.9)$$

La formule 4.15 (chapitre 4, page 47) devient :

$$\begin{aligned} x \xleftarrow{HELLO_A} A, x \xleftarrow{HELLO_B} B, \\ \forall y, (@IP_y, sym) \in HELLO_A \text{ ou } (@IP_y, mpr) \in HELLO_A : \\ (@IP_y, sym) \in HELLO_B \text{ ou } (@IP_y, mpr) \in HELLO_B, \\ \exists @IP_z \in TC_A \cap TC_B \\ \Downarrow \\ x \neg trusts(\{IdP_A, IdP_B, IdP_z\}) \end{aligned} \quad (A.10)$$

La formule 4.21 (chapitre 4, page 53) devient :

$$\begin{aligned}
T &= (IdP_y, IdP_B, N, I) \in RT_x, \exists IdP'_B \in NS_x, \\
x &\xleftarrow{DATA_{y-x}} B' : hop_count(DATA_{y-x}) \neq N \\
&\Downarrow \\
&\forall w \in route_{y \rightarrow B', x-trusts}(IdP_w)
\end{aligned} \tag{A.11}$$

La formule 4.22 (chapitre 4, page 54) devient :

$$\begin{aligned}
T' &= (IdP_x, IdP_C, N_p, I') \in RT_y, \exists IdP'_C \in NS_y, \\
y &\xleftarrow{DATA_{x-y}} C' : hop_count(DATA_{x-y}) \neq N_p \\
&\Downarrow \\
&\forall w \in route_{x \rightarrow C', y-trusts}(IdP_w)
\end{aligned} \tag{A.12}$$

Annexe B

Présentation de GloMoSim

GloMoSim simule des réseaux avec plus de mille nœuds reliés par une capacité de communications hétérogènes comme le multicast, le multi-saut en utilisant les communications sans fil ad-hoc, et les protocoles Internet traditionnelles. Le tableau suivant dresse la liste des modèles actuellement disponibles dans GloMoSim à chacune des grandes couches :

Couche	Modèle
Physique	Free space, Two-Ray
MAC	CSMA, MACA, TSMA, 802.11
Réseaux (routage)	Bellman-Ford, FSR, OSPF, DSR, WRP, LAR, AODV
Transport	TCP, UDP
Application	Telnet, FTP

B.1 Installation

GloMoSim est disponible gratuitement pour les universitaires dans le site officiel [glo], il est multi-platform et requiert l'installation d'un compilateur C (*gcc* sous linux et *Visual C++* sous windows). Nous avons travaillé dans un environnement Linux sous Ubuntu-8.04 (Hardy Heron), l'installation de GloMoSim se déroule selon les étapes suivantes :

1. Décompresser le fichier téléchargé en utilisant la commande suivante :

```
tar xzf ../glomomosim-2.03.tar.gz
```

Après décompression du fichier téléchargé, deux sous-répertoires sont trouvés : *glomosim* et *parsec*. Plusieurs bibliothèques PARSEC pré-compilés sont disponibles dans le répertoire *parsec* pour différents systèmes d'exploitation (Windows, Red-Hat, Solaris...etc).

2. Installer la variable d'environnement *PCC_DIRECTORY* dans le répertoire de *parsec* :

```
$ cd glomosim/glomosim-2.03
$ export PCC_DIRECTORY='pwd'/parsec/redhat-7.2
```

3. Ajouter le répertoire du compilateur pcc (Parsec) dans la variable d'environnement PATH :

```
$ cd glomosim/glomosim-2.03/parsec/redhat-7.2/bin/
$ export PATH=$PATH:'pwd'
```

4. Compiler GloMoSim à l'aide de la commande suivante :

```
$ cd glomosim/main
$ make
```

5. Lancer un test avec la commande suivante :

```
$ cd glomosim/bin
$ ./glomosim config.in
```

Les messages suivants doivent apparaître :

```
$ ./glomosim config.in
Node 0 (280.47, 212.84, 0.00).
Node 1 (639.89, 179.14, 0.00).
Node 2 (720.11, 198.10, 0.00).
Node 3 (1211.61, 216.94, 0.00).
Node 4 (1415.87, 23.07, 0.00).
Node 5 (1980.56, 119.08, 0.00).
Node 6 (214.29, 472.47, 0.00).
Node 7 (636.93, 455.61, 0.00).
Node 8 (985.39, 574.96, 0.00).
Node 9 (1023.64, 484.49, 0.00).
Partition 1 (0 2) has range (0, 0) to (0, 0): 10 nodes
Current Sim Time[s] = 0.000000000 Real Time[s] = 0 Completed 0%
Current Sim Time[s] = 9.073651536 Real Time[s] = 0 Completed 1%
Current Sim Time[s] = 18.014513152 Real Time[s] = 0 Completed 2%
.....
Current Sim Time[s] = 873.064564290 Real Time[s] = 1 Completed 97%
Current Sim Time[s] = 882.072024056 Real Time[s] = 1 Completed 98%
Current Sim Time[s] = 891.081945395 Real Time[s] = 1 Completed 99%
Execution time : 1.1201 sec
Number of events (including timeouts) processed : 605409
Number of messages processed : 4
Number of context switches occurred : 12
Number of Local NULL messages sent : 0
Number of Remote NULL messages sent : 0
Total Number of NULL messages sent : 0
NULL messages / Regular messages : 0.000
```

6. A la fin de la simulation, les statistiques sont générées dans un fichier appelé *GLOMO.STAT* sur le répertoire *BIN*.

B.2 Configuration

Comme présenté précédemment, la simulation commence par l'exécution de la commande suivante :

```
$ ./glomosim < input file >
```

Le fichier <input file> contient les principaux paramètres de simulation d'un scénario particulier, par exemple : le nombre de nœuds, le temps de simulation, le protocole utilisé,...etc, Un exemple de ce fichier est disponible dans le répertoire *BIN* (config.in).

Le tableau suivant récapitule les paramètres les plus utilisés :

Paramètre	Description
SIMULATION-TIME	Le temps maximum de simulation. Exemple : 100NS (100 nano-secondes), 100MS (100 milli-secondes), 100S ou 100 (100 secondes), 100M (100 minutes), 100H (100 heures) et 100D (100 jours).
SEED	Nombre aléatoire utilisé pour initialiser différents paramètres aléatoires pour la simulation.
TERRAIN-DIMENSIONS	Dimension du terrain de simulation exprimée en mètres.
NUMBER-OF-NODES	Nombre de nœuds du scénario à simuler.
NODE-PLACEMENT	Représente la technique de placement des nœuds, et peut avoir les valeurs suivantes : RANDOM (placé aléatoirement), UNIFORM (selon le nombre des nœuds, le terrain sera divisé en un ensemble de cellules, et dans chaque cellule les nœuds sont placés aléatoirement), GRID (le placement des nœuds commence à la position 0,0 et placés dans une grille, où chaque nœud est placé avec une distance de GRID-UNIT de ses voisins), et FILE (la position des nœuds est défini dans le fichier NODE-PLACEMENT-FILE).
MOBILITY	Représente le modèle de mobilité, par exemple : NONE (sans mobilité), RANDOM-WAYPOINT (mobilité aléatoire). Dans ce modèle, le nœud sélectionne aléatoirement une destination, et se déplace en direction de cette destination avec une vitesse entre MOBILITY-WP-MIN-SPEED et MOBILITY-WP-MAX-SPEED. Quand il atteint cette destination, le nœud reste dans cette position pendant la durée définis par MOBILITY-WP-PAUSE.
NETWORK-PROTOCOL	Définit le protocole réseaux, par exemple : IP.
ROUTING-PROTOCOL	Définit le protocole de routage (exemple :DSR, OLSR).

Les valeurs par défaut de ses paramètres dans le fichier de configuration CONFIG.IN sont :

SIMULATION-TIME	15M
SEED	1
TERRAIN-DIMENSIONS	(2000, 2000)
NUMBER-OF-NODES	30
# NODE-PLACEMENT	FILE
# NODE-PLACEMENT-FILE	./nodes.input
# NODE-PLACEMENT	GRID
# GRID-UNIT	30
# NODE-PLACEMENT	RANDOM
NODE-PLACEMENT	UNIFORM
MOBILITY	NONE

Il est important de noter que les lignes précédés par le signe # sont considérées comme des commentaires. Un exemple de placement des nœuds est présenté dans le fichier NODES.INPUT du répertoire BIN :

Format: nodeAddr 0 (x, y, z)
Le deuxième paramètres est pour la compatibilité
avec le format de trace de mobilité
0 0 (20 , 0.5, 0.21)
1 0 (12.5, 30.8, 0.08)
2 0 (20.4, 60.7, 0.11)

Finalement, les paramètres suivants détermine si on a besoin des résultats statistiques de différentes couches. En initialisant les paramètres à *YES*, la simulation fournit les statistiques de la couche correspondantes.

APPLICATION-STATISTICS	YES
TCP-STATISTICS	NO
UDP-STATISTICS	NO
ROUTING-STATISTICS	NO
NETWORK-LAYER-STATISTICS	NO
MAC-LAYER-STATISTICS	NO
RADIO-LAYER-STATISTICS	NO
CHANNEL-LAYER-STATISTICS	NO
MOBILITY-STATISTICS	NO
# GUI-OPTION: YES	Permet à GloMoSim de communiquer
#	avec l'outil de visualisation
GUI-OPTION	YES
GUI-RADIO	YES
GUI-ROUTING	YES

Glossaire

M-SOLSR : Message based secure OLSR (la version sécurisé de OLSR proposé par [ACJ⁺03])

P-SOLSR : Paquet based secure OLSR (la version sécurisé de OLSR proposé par [HTR⁺04])

Bibliographie

- [ACJ⁺03] C. Adjih, T. Clausen, P. Jacquet, P. Mühlethaler, A. Laouiti, and D. Raffo. Securing the OLSR protocol. In *Med-Hoc-Net*, June 25-27 2003.
- [AD01] K. Aberer and Z. Despotovic. Managing trust in a peer-2-peer information system. In *Proceedings of the Tenth International Conference on Information and Knowledge Management (CIKM01)*. ACM Press, 2001.
- [AdSJBM07] Asmaa Adnane, Rafael Timoteo de Sousa Jr, Christophe Bidan, and Ludovic Mé. Analysis of the implicit trust within the OLSR protocol. In *IFIPTM07 : Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, volume 238 of *IFIP International Federation for Information Processing*. Springer, july 2007. ISBN : 978-0-387-73654-9.
- [Aur05] T. Aura. Cryptographically generated addresses (CGA). *Request for Comments : 3972*, 2005.
- [BB02] Sonja Buchegger and Jean-Yves Le Boudec. Performance Analysis of the CONFIDANT Protocol (Cooperation Of Nodes - Fairness In Dynamic Ad-hoc NeTworks). In *MobiHoc 2002 : The Third ACM International Symposium on Mobile Ad Hoc Networking and Computing*, 2002.
- [BCSC05] Ciaràn Bryce, Paul Couderc, Jean-Marc Seigneur, and Vinny Cahill. Implementation of the SECURE trust engine. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, *Trust Management, Third International Conference, iTrust 2005, Paris, France, May 23-26, 2005, Proceedings*, volume 3477 of *Lecture Notes in Computer Science*, pages 397–401. Springer, 2005.
- [BDK⁺05] Ciaràn Bryce, Nathan Dimmock, Karl Krukow, Jean-Marc Seigneur, Vinny Cahill, and Waleed Wagealla. Towards an evaluation methodology for computational trust systems. In Peter Herrmann, Valérie Issarny, and Simon Shiu, editors, *Trust Management, Third International Conference, iTrust 2005, Paris, France, May 23-26, 2005, Proceedings*, volume 3477 of *Lecture Notes in Computer Science*, pages 289–304. Springer, 2005.
- [BFL96] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *IEEE Symposium on Security and Privacy*, Oakland, CA, 1996. IEEE Computer Society Press.

- [BH01] L. Buttyan and J. Hubaux. Nuglets : a virtual currency to stimulate cooperation in self-organized ad hoc networks. Technical Report DSC/2001, 2001.
- [BTA⁺] Lokesh Bajaj, Mineo Takai, Rajat Ahuja, Ken Tang, Rajive Bagrodia, and Mario Gerla. Technical report.
- [CGS⁺03] V. Cahill, E. Gray, J. Seigneur, C. D. Jensen, Y. Chen, B. Shand, N. Dimmock, A. Twigg, J. Bacon, C. English, W. Wagealla, S. Terzis, P. Nixon, G. di Marzo Serugendo, C. Bryce, M. Carbone, K. Krukow, and M. N. Using trust for secure collaboration in uncertain environments. *IEEE Pervasive Computing*, 2(3) :52–61, jul 2003.
- [CH07] Eddy Cizeron and Salima Hamma. A multiple description coding strategy for multi-path in mobile ad hoc networks. *ICLAN : Second International Conference on the Latest Advances in Networks*, Decembre 2007.
- [CJ03] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol OLSR. *Requet for Comments : 3626, IETF*, October 2003.
- [CLM⁺05] Thomas Clausen, Anis Laouiti, Paul Muhlethaler, Daniele Raffo, and Cédric Adjih. Securing the OLSR routing protocol with or without compromised nodes in the network. Technical report, HIPERCOM Project, INRIA Rocquencourt, february 2005.
- [CR07] Cuppens-Boulahia N. Nuon S. Cuppens, F. and T. Ramard. Property based intrusion detection to secure OLSR. In *ICWMC '07 : Proceedings of the Third International Conference on Wireless and Mobile Communications, IEEE Computer Society*, pages 52–52, 2007.
- [CS99] Macker J. Corson S. Mobile ad hoc networking (MANET), routing protocol performance issue and evaluation consideration. *Requet for Comments : 2501, IETF*, 1999.
- [Deu62] Morton Deutsch. Cooperation and trust : Some theoretical notes. In *Jones, M. R. (ed.), Nebraska Symposium on Motivation. Nebraska University Press*, 1962.
- [DH98] S. Deering and R. Hinden. Internet protocol, version 6 (IPv6) specification. *Internet Engineering Task Force (IETF), RFC2460*, December 1998.
- [DLRS02] B. Dahill, B. Levine, E. Royer, and C. Shields. A secure routing protocol for ad hoc networks. In *ICNP*. IEEE Computer Society, 2002.
- [dSJABM07] Rafael Timoteo de Sousa Jr, Asmaa Adnane, Christophe Bidan, and Ludovic Mé. Analyse de la confiance implicite requise dans le routage adhoc OLSR. In *SARSSI'07 : The 2nd joint conference on security in network architectures and information systems*, june 2007. ISBN : 2-916377-49-2.
- [Gam80] Diego Gambetta. *Can We Trust Trust?*, chapter Trust : Making and Breaking Cooperative Relations, pages 213–237. Department of Sociology, University of Oxford, 1980.

- [GHG02] M. Gerla, X. Hong, and G. Pei. Fisheye state routing protocol (FSR) for ad hoc networks. *INTERNET-DRAFT, IETF*, 2002.
- [glo] GloMoSim : Global mobile informations systems simulation library. <http://pcl.cs.ucla.edu/projects/glomosim/>.
- [GS00] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys and Tutorials*, 2000.
- [HP04] Yih-Chun Hu and Adrian Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy*, 2(3) :28–39, 2004.
- [HPJ02] Y. Hu, A. Perrig, and D. Johnson. Ariadne : A secure on-demand routing protocol for ad hoc networks, 2002.
- [HTR⁺04] Andreas Hafslund, Andreas Tønnesen, Roar Bjørgum Rotvik, Jon Andersson, and Øivind Kure. Secure extension to the OLSR protocol. In *OLSR Interop and Workshop*, August 2004.
- [IETF] IETF : The internet engineering task force, <http://www.ietf.org/>.
- [JHM03] D. Johnson, Y. Hu, and D. Maltz. The dynamic source routing protocol for mobile ad hoc networks (dsr). *Request for Comments : 4728, IETF*, February 2003.
- [Joh73] D. B. Johnson. A note on dijkstra’s shortest path algorithm. *Journal of the ACM*, pages 385–388, jul 1973.
- [KLMC08] Amir R. Khakpour, Maryline Laurent-Maknavicius, and Hakima Chaouchi. WATCHMAN : An overlay distributed AAA architecture for mobile ad hoc networks. In *ARES*, pages 144–152. IEEE Computer Society, 2008.
- [LBMA04] Anis Laouiti, Saadi Boudjit, Pascale Minet, and Cédric Adjih. OLSR for IPv6 networks. In *Med Hoc Net*, Juin 2004.
- [LBP07] Jerome Lebegue, Christophe Bidan, and Thierry Plesse. An OLSR extension to deal with predefined groups. In *Mobile Summit*, July 2007.
- [LBPB06] Jerome LEBEGUE, Christophe Bidan, Thierry PLESSE, and David BOUCART. Vers une gestion de groupe sécurisée dans les réseaux ad hoc militaires. In *SAR/SSI*, juin 2006.
- [Mar94] S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, 1994.
- [MC02] C. Montenegro and C. Castelluccia. Statistically unique and cryptographically verifiable (SUCV) identifiers and addresses. In *NDSS-02*, 2002.
- [MGLB00] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.
- [MM02] Pietro Michiardi and Refik Molva. Core : a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks. In *In proceedings of IFIP Communications and Multimedia Security Conference (CMS)*, 2002.

- [MVU06] K. Meka, M. Virendra, and S. Upadhyaya. Trust based routing decisions in mobile ad-hoc networks. In *Proceedings of the Workshop on Secure Knowledge Management*, 2006.
- [ND01] T. Narten and R. Draves. Privacy extensions for stateless address auto-configuration in IPv6. *Request For Comment : 3041*, January 2001.
- [nii] OLSR patch for GlomoSim. <http://www.net.ie.niigata-u.ac.jp/mase/olsr/>.
- [NS2] NS-2 : Network simulator. <http://www.isi.edu/nsnam/ns/>.
- [NS03] Peng Ning and Kun Sun. How to misuse AODV : A case study of insider attacks against mobile adhoc routing protocols,. In *in Proceedings of the 4th Annual IEEE Information Assurance Workshop*,, pages 60–67, 2003.
- [OR01] G. O’Shea and M. Roe. Child-proof authentication for MIPv6 (cam). In *ACM SIGCOMM Computer Communication Review*, pages 4–8, 2001.
- [Pal] Nicolas prigent and olivier heen and jean-pierre andreaux and christophe bidan. secure distributed system for management of local community representation within network devices. patent : ep1614269, janvier 2006.
- [PB94] C. E. Perkins and P. Bhagwat. Highly-dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings, 1994 SIGCOMM Conference*, pages 234–244, 1994.
- [PBAH03] Nicolas Prigent, Christophe Bidan, Jean-Pierre Andreaux, and Olivier Heen. Secure long term communities in ad hoc networks. In Sanjeev Setia and Vipin Swarup, editors, *Proceedings of the 1st ACM Workshop on Security of ad hoc and Sensor Networks, SASN 2003, Fairfax, Virginia, USA, 2003*, pages 115–124. ACM, 2003.
- [PBRD03] C. Perkins, E. Belding-Roye, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. *Requet for Comments : 3561, IETF*, July 2003.
- [PMdSJ04] R. S. Puttini, L. Mé, and R. T. de Sousa Jr. On the vulnerabilities and protection of mobile ad hoc network routing protocols. In *Proceedings of the 3rd International Conference on Networking ICN’2004*, 2004.
- [PPMdS04] Ricardo Staciarini Puttini, Jean-Marc Percher, Ludovic Mé, and Rafael de Sousa. A fully distributed IDS for MANET. In *ISCC*, pages 331–338. IEEE Computer Society, 2004.
- [RACM04] Daniele Raffo, Cédric Adjih, Thomas Clausen, and Paul Mühlethaler. An advanced signature system for OLSR. In *in SASN-04 : Proceedings of the 2nd ACM Workshop on security of ad hoc and sensor networks. New York, NY, USA : ACM Press*, pages 10–16, 2004.
- [RACM05] Daniele Raffo, Cédric Adjih, Thomas Clausen, and Paul Mühlethaler. Securing OLSR using node locations. In *Proceedings of 2005 European Wireless (EW 2005)*, pages 437–443, April 10–13 2005.

- [RK05] S. Ruohomaa and L. Kutvonen. Trust management survey. In *Trust Management, Third International Conference, iTrust 2005, Paris, France*. Springer, 2005.
- [sim] Simulation vs. emulation : Evaluating mobile ad hoc network routing protocols.
- [VB07] J.P. Vilela and J. Barros. A feedback reputation mechanism to secure the optimized link state routing protocol. In *3rd IEEE/CreateNet International Conference on Security and Privacy in Communication Networks*, 2007.
- [VG03] F. Dupont S. Gombault B. Tharon V. Gayraud, L. Nuaymi. La sécurité dans les réseaux sans fil ad hoc. *SSTIC (Symposium sur la Sécurité des Technologies de l'Information et de la Communication)*, Rennes, France, 2003.
- [Vil05] L. Viljanen. Towards an ontology of trust. In *Trust, Privacy and Security in Digital Business*. Springer, 2005.
- [WIK] Optimized link state routing protocol. <http://wiki.uni.lu/secan-lab/optimized+link+state+routing+protocol.html>.
- [WLMG05] M. Wang, L. Lamont, P Mason, and M. Gorlatova. An effective intrusion detection approach for OLSR MANET protocol. In *First Workshop on Secure Network Protocols (NPsec)*. Boston, Massachusetts, USA, July 2005.
- [XRM04] Li Xiaoqi, Lyu Rung, and Tsong Michael. A trust model based routing protocol for secure ad hoc networks. In *Proceedings 2004 IEEE Aerospace Conference*, pages 1286– 1295, March 2004.
- [YKB93] R. Yahalom, B. Klein, and T. Beth. Trust relationships in secure systems-A distributed authentication perspective. In *Proceedings of the IEEE Symposium on Security and Privacy*, 1993.
- [YNK01] S. Yi, P. Naldurg, and R. Kravets. Security-aware ad hoc routing for wireless networks. In *Proceedings of the 2nd ACM international symposium on mobile ad hoc networking and computing*, pages 299–302, 2001.
- [Zap05] Manel Guerrero Zapata. Secure ad hoc on-demand distance vector (SAODV) routing. *IETF MANET Internet Draft*, March 2005.
- [ZBG] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. Glomosim : a library for parallel simulation of large-scale wireless networks.
- [ZLH00] Y. Zhang, W. Lee, and Y. Huang. Intrusion detection techniques for mobile wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking (MOBICOM)*, pages 275–283. ACM Press, August 2000.

Table des figures

2.1	Routage optimisé avec OLSR	8
2.2	Découverte de voisinage par les nœuds OLSR	9
2.3	Découverte de voisins à 2 sauts	9
2.4	Génération et retransmission des messages TC	10
2.5	Vision possible du nœud OLSR sur chaque nœud de son voisinage à 2 sauts	11
2.6	Découverte de routes dans AODV	13
3.1	Chaîne de confiance entre les nœuds MPR	31
3.2	Diagramme de séquence de signalement des MPR	32
3.3	Propagation de la confiance entre les nœuds MPR	35
3.4	Fabrication de message HELLO	36
4.1	Exemple de deux nœuds x et y ayant le même voisinage	46
4.2	Exemple d'un nœud y ayant le voisinage inclus dans celui de x	47
4.3	Exemple illustrant la propriété du plus court chemin	48
4.4	Modèle d'attaque sur la topologie du réseau	50
4.5	Fabrication de message HELLO	56
5.1	Format du paquet M -SOLSR	60
5.2	Format de la signature dans SOLSR	61
5.3	Format du paquet OLSR	61
5.4	Exemple des échanges dans M -SOLSR	62
5.5	Format de la signature dans P -SOLSR	63
5.6	Format du paquet P -SOLSR	63
5.7	Exemple des échanges dans P -SOLSR	64
5.8	Exemple des échanges avec l'identité prouvable et M -SOLSR	68
5.9	Exemple de détection d'attaque et envoi d'une preuve	73
5.10	Distribution de la preuve de méfiance (alerte)	74
6.1	Exemple de scénario : A est l'attaquant, et T est la cible	80
6.2	Etablissement de l'attaque	81
6.3	Détection de l'attaque	82
6.4	Taux de détection : 50 nœuds, un attaquant et une cible	83
6.5	Taux de détection : 50 nœuds, un attaquant et deux cibles	83

6.7	Exemple où la cible est isolée du réseau par l'attaquant	84
6.6	Taux de détection : 50 nœuds, deux attaquants et une cible chacun	84
6.8	Vision de la topologie après la détection de l'attaque	86
6.9	Diagramme représentant le taux de faux positifs selon différentes vitesses	89
6.10	Taux de faux positifs par rapport à la pause de déplacement	89

Résumé

La notion de confiance, quoique implicite, est toujours présente dans le fonctionnement des protocoles, en particulier, entre les entités qui participent aux opérations de routage. Dans notre travail, nous nous sommes intéressés à la gestion de la confiance (*trust management*) comme une solution de sécurité pour le protocole OLSR (Optimized Link State Routing Protocol). Cette approche s'adapte particulièrement à la nature mobile, ad-hoc, distribuée et auto-organisée des réseaux ad-hoc. De plus, la gestion explicite de la confiance permet aux entités de raisonner avec et à propos de la confiance, les rendant ainsi plus robustes pour la prise de décisions concernant les autres entités.

Nous commençons par une analyse du fonctionnement du protocole OLSR sous l'optique de la confiance. Ensuite, nous proposons un raisonnement basé sur la confiance pour permettre à chaque nœud d'évaluer le comportement des autres nœuds, de détecter les comportements malveillants, et donc de pouvoir décider de faire confiance ou non. Enfin, nous proposons une solution de prévention et des contremesures pour résoudre le cas d'une incohérence et contrer les comportements malveillants. Ces solutions ne nécessitent que peu de modifications sur le protocole OLSR et peuvent être étendues selon le type d'attaque et les besoins des utilisateurs.

Mots-Clés : sécurité, routage, réseaux ad-hoc, MANet, OLSR, sécurité du routage, confiance, gestion de la confiance.

Abstract

The implicit trust is always present in the operating protocols, in particular, between the entities involved in routing operations. In our work, we are interested by trust management as security solution for OLSR protocol. This approach fits particularly with characteristic of ad-hoc networks. Moreover, the explicit trust management allows entities to reason with and about trust, and to take decisions regarding other entities. First, we analyse the functioning of OLSR protocol under the terms of trust. Then, we propose a trust-based reasoning which allow to each node to evaluate the behavior of the other nodes, to detect misbehavior nodes, and then to be able to trust or not the other nodes. Finally, we propose a solutions of prevention and contermesures to resolve the situations of inconsistency, and counter the malicious nodes. These solutions require only minor modifications on the OLSR and can be extended depending on the attack type and users needs.

Keywords : security, routing protocol, ad-hoc network, MANet, OLSR, routing security, trust.

VU :
Le Directeur de Thèse
Ludovic MÉ

VU :
Le Responsable de l'École Doctorale
Olivier BONNAUD

VU pour autorisation de soutenance
Rennes, le

Le Président de l'Université de Rennes 1

Guy CATHELINÉAU

VU après soutenance pour autorisation de publication :
Le Président du Jury,