Mécanismes de Sécurité pour des Protocoles de Routage des Réseaux ad hoc

Xiaoyun Xue

Thèse

présentée pour obtenir le grade de Docteur
de l'École Nationale Supérieure des Télécommunications

Spécialité : **Informatique et Réseaux**

Xiaoyun Xue

Mécanismes de Sécurité pour des Protocoles de Routage des Réseaux Ad hoc

Soutenue le 29 Septembre 2006 devant le jury composé de :

Bijan Jabbari	Université de George Mason	Président
Abdelmajid Bouabdallah	Université de Compiègne	Rapporteur
Paul Mühlethaler	INRIA Rocquencourt	Rapporteur
Farid Naït-Abdesselam	Université de Lille	Examinateur
Jean Leneutre	ENST Paris	Examinateur
Jalel Ben-Othman	Université de Versailles	Examinateur
Michel Riguidel	ENST Paris	Directeur de thèse

PhD Thesis

École Nationale Supérieure des Télécommunications

Computer Science and Network Department

Xiaoyun Xue

Security mechanisms for ad hoc routing protocols

Defense date: Septembre, 29th 2006

Committee in charge:

Bijan Jabbari	George Mason University	President
Abdelmajid Bouabdallah	University of Compiègne	Reporter
Paul Mühlethaler	INRIA Rocquencourt	Reporter
Farid Naït-Abdesselam	University of Lille	Examinator
Jean Leneutre	ENST Paris	Examinator
Jalel Ben-Othman	University of Versailles	Examinator
Michel Riguidel	ENST Paris	Thesis director

To my family and my friends,
À ma famille et mes amis,

Acknowledgments

During the preparation of this thesis, I have been accompanied and supported by many people who, I believe, deserve this honor more than myself. I feel relieve to have the opportunity to express my gratitude for all of them.
I am greatly indebted to my supervisor Jean Leneutre for all his helps and his preciseness research ethos. Many thanks for his suggestions, thoughts, and constructive criticisms during the past four years, and many special thanks for his patience and encouragement. I have learned much from him.
I would like to express my sincere thankfulness to my supervisor Jalel Ben-Othman. He accompanied me since my DEA in the University of Versailles, and our cooperation was fruitful and pleasant.
I am grateful to my director Michel Riguidel. He has been keeping a close eye on the progress of my work from the very beginning of this thesis. Every discussion with him was educational and constructive.
I wish equally to express my gratitude without reservation to Lin Chen. His work helped me a lot during the end of my thesis preparation, and it was a pure pleasure to work with him.
It is also for me a great opportunity to deliver my gratefulness to all professors who accepted to be in the jury of my thesis defense, and other professors like Ahmed Serhrouchni, Ahmed Mehaoua. Thanks also to my old professors in China: NianJun Zhang, HuanGuo Zhang, LinLin Wu, KangMing Ong, etc.
I would like to thank all my old and new friends at the INFRES department of ENST and at the PRiSM laboratory. Their wonderful friendship will certainly be always kept in my mind.
I want to thank my old friend Xia Cao who had visited me during my stay at Paris, I cannot forget our discussion until three o'clock in the morning.
Last but certainly not least, I loudly thank all the members in my familly : my husband Xuewen Wang who has always tried his best to support me, my sister Xiaohui Xue who has always been very close to me to give me suggestions and tenderness, my brother-in-law Xuewu Wang who had corrected a part of this thesis, and my caring parents who always give me encouragement and solicitude.

Résumé

Introduction générale

Contexte

Avec l'émergence des équipements mobiles, et la prolifération des points d'accès de réseaux, les utilisateurs en déplacement ont aujourd'hui de plus en plus de possibilités d'accéder aux réseaux et aux informations. À l'avenir, il serait même possible de se connecter à tout type de réseaux en utilisant tout type d'appareils, à tout moment et en tout lieu. Ce nouveau paradigme de réseaux de est appelé *réseaux de quatrième génération* où les réseaux filaires et sans-fil, terrestres et satellites seront connectés ensemble pour former un grand réseau universel. Les utilisateurs auront une mobilité maximale grâce aux handovers verticaux entre les différents réseaux, et une qualité de service optimale grâce à la nouvelle architecture et les nouveaux mécanismes des réseaux.
Les réseaux ad hoc (aussi appelés MANETs pour "Mobile Ad hoc NETworks") sont des réseaux spéciaux qui apparaissent dans ce contexte. Étant auto-organisé et autonome, un réseau ad hoc peut être soit une partie du réseau universel, soit indépendant et contenant uniquement des nœuds mobiles qui communiquent entre eux sans aucune infrastructure. Ainsi, les MANETs peuvent être déployés à la demande dans des locaux lointains ou des périmètres physiquement délimités.
Ayant une facilité de déploiement et un faible coût, les MANETs peuvent être utilisés pour plusieurs scénarios. Par exemple, les champs de bataille, conférences, services d'urgence, etc.
Les MANETs sont initialement développés pour des applications militaires, y compris des activités de sauvetages quand les moyens de communication basés sur une infrastructure conventionnelle sont détruits par guerre, catastrophe naturelle, etc. Autrement, ils peuvent aussi être utilisés dans des zones résidentielles fournissant une façon de communication supplémentaire pour des utilisateurs mobiles au long d'une autoroute ou dans un campus universitaire, etc. À noter aussi que les divers scénarios et applications sont différents sur nombreux aspects, en particulier sur la dimension du réseau, la capacité des nœuds, l'hostilité d'environnement, l'exigence en terme de service et de sécurité, etc.
Le routage ad hoc est très différent de celui des réseaux traditionnels. Dans les réseaux traditionnels comme Internet ou réseaux cellulaires, ce sont des routeurs dédiés qui prennent en charge de sauvegarder et de transférer les données pour les nœuds terminaux. Tandis que dans les réseaux ad hoc, puis qu'il n'existe pas de routeur dédié, le routage doit être effectué par chacun des nœuds dans ces réseaux pour assurer une disponibilité maximale de service de routage. Ainsi, tout nœud est à la fois terminal et routeur, et il doit échanger avec d'autres nœuds non seulement

du trafic d'applications, mais aussi des messages pour le contrôle de réseau et du routage. De plus, le changement de topologie, le partitionnement de réseau, le taux élevé d'erreur de transmission, interférences et collisions, la limite de bande passante et d'énergie sont des problèmes à considérer dans la conception de protocoles de routage pour les réseaux ad hoc.

Besoins de sécurité du routage ad hoc

La sécurité est un sujet important à traiter, surtout pour les applications de MANET dites *sensibles à la sécurité* (par exemple une application du type champ de bataille). Effectivement, les réseaux ad hoc indépendants sont connus pour leur manque d'organisation, de planning et de configuration, donc ils sont généralement considérés difficiles à sécuriser.
Nous avons déjà signalé que le routage ad hoc est très différent de celui des réseaux traditionnels. Il en est de même pour sa sécurité. Concrètement, pour sécuriser le routage dans un réseau traditionnel, il est suffisant de protéger et d'authentifier les routeurs dédiés (sous l'hypothèse que les nœuds expéditeurs et destinataires sont bienveillants), mais pour assurer la sécurité du routage dans un réseau ad hoc, chacun des nœuds doit non seulement prendre la responsabilité de ses propres comportements mais aussi vérifier les comportements des autres nœuds.
Le travail réalisé dans le cadre de cette thèse se focalise sur la sécurité du routage ad hoc. Nous allons dans un premier temps discuter les raisons pour lesquelles les mécanismes de sécurité conçus pour les réseaux traditionnels (filaires et cellulaires) ne sont pas adaptés aux réseaux ad hoc, et les nouveaux besoins de la sécurité du routage ad hoc :

Raison 1 : nœuds compromis. Les nœuds dans un MANET sont plus faciles à compromettre que ceux dans un réseau traditionnel, parce qu'ils sont de nature mobile et sans-fil donc physiquement plus petits et plus faciles à déplacer et à attaquer. De plus, parce qu'ils peuvent éventuellement entrer et/ou sortir du réseau de temps à autre, et que les réseaux ad hoc peuvent être divisés et/ou fusionnés, les attaquants auront plus de chances d'attaquer (compromettre) des nœuds sans être aperçus.

Malheureusement, il y a quelques attaques très sophistiquées, par exemple les attaques de type "wormhole" où des nœuds compromis attaquent en coopérant, qui ne peuvent être commises que par des nœuds compromis et sont difficiles à éviter.

Nouveau besoin : parce que les nœuds compromis ne peuvent pas être détectés par simple authentification, ce problème ne peut pas être résolu par l'utilisation de cryptographie. Donc, nous devons considérer spécialement d'autres solutions pour ce problème.

Raison 2 : faible capacité, ou nœuds hétérogènes. La capacité souvent limitée des nœuds et l'utilisation de batteries pour l'alimentation des équipements

sont aussi des faiblesses des réseaux ad hoc. Les nœuds ad hoc peuvent ainsi avoir une durée de vie limitée. De plus, pour gagner plus de ressources, des nœuds peuvent aussi être "gourmands", par exemple vouloir gagner plus de bande passante.

Nouveau besoin : puisqu'ils ont été plutôt désignés pour les nœuds physiquement plus forts, les mécanismes de sécurité des réseaux traditionnels sont inadaptés à l'environnement des réseaux ad hoc. Les MANETs ont donc besoin de nouvelles solutions de sécurité qui doivent être économes en terme de puissance de calcul, de consommation d'énergie et de la charge ("overhead") du trafic. En outre, ces nouveaux mécanismes doivent aussi être équitables au niveau de l'utilisation de ressources du réseau.

Raison 3 : manque de coopération. Parce que les nœuds dans un réseau ad hoc ont tendance à être égoïstes à cause du manque de ressource, nous devons assurer la coopération entre eux. Malheureusement, il est difficile de détecter des nœuds égoïstes : les nœuds peuvent tout simplement être silencieux et/ou refuser de transférer les données. Quand de tels nœuds sont nombreux dans le réseau, la disponibilité du service de routage est atteinte.

Nouveau besoin : normalement, le problème d'égoïsme n'existe pas dans les réseaux traditionnels où les nœuds ne dépendent pas des autres mais se reposent sur les routeurs dédiés pour assurer la fonctionnalité du routage. Donc, de nouveaux mécanismes doivent être désignés pour garantir la coopération des nœuds dans des réseaux ad hoc.

Raison 4 : manque d'organisation. Le manque d'organisation influence elle aussi la sécurité des MANETs. Parce qu'un nœud n'a pas forcément de connaissance sur les autres lors de la montée du réseau, la confiance a-priori peut ne pas exister. De plus, parce qu'il n'y a pas forcément de serveur central, la distribution et la gestion (surtout la révocation) de clés peuvent être difficiles à réaliser. D'autre part, à cause de la dynamicité du réseau, il n'est pas facile de gérer l'adhésion des membres du réseau. Tous ces problèmes génèrent de sérieuses difficultés pour la sécurité du routage ad hoc.

Nouveau besoin : les solutions de sécurité pour les réseaux traditionnels s'appuient souvent sur des relations de confiance préalablement établies ou des autorités de confiance à tierces. Elles utilisent les primitives cryptographiques symétriques et/ou asymétriques pour authentifier les nœuds et sécuriser les échanges de données. Afin d'utiliser ces moyens cryptographiques dans les MANETs, nous devons étudier comment établir des autorités de confiance et/ou des relations de confiance entre les nœuds sans l'aide d'aucune infrastructure.

Raison 5 : mobilité. La mobilité des nœuds rend la topologie des MANETs instable. Il n'est donc pas facile pour un nœud de connaître correctement son voisinage et la topologie du réseau. Les attaquants peuvent ainsi forger et diffuser des fausses informations de topologie pour réaliser leurs attaques. Par

ce moyen, un protocole de routage ad hoc non-sécurisé peut facilement être attaqué. De plus, la mobilité des attaquants peut aussi les rendre plus difficiles à détecter ou localiser.

Nouveau besoin : il n'y a pas autant de mobilité dans les réseaux filaires. De plus, dans les réseaux cellulaires ce sont des infrastructures qui gèrent la mobilité. Donc, des protocoles de routage capables de découvrir correctement la topologie du réseau même sous attaques doivent être conçus spécialement pour les MANETs.

Raison 6 : interface sans-fil (radio). L'interface sans-fil (dans la plus part des cas l'interface radio) des nœuds pose aussi des problèmes dans le routage ad hoc. À cause de la nature de radio en transmission qui est la diffusion, chaque paquet émit dans le réseau, que ce soit en unicast ou en diffusion, pourrait être reçu par tout voisin de son émetteur. De plus, le *problème des nœuds cachés*, où deux émetteurs qui ne peuvent pas entendre l'un à l'autre envoient à un même récepteur en même temps, peut causer collisions. En outre, le *problème de nœud exposé*, où les nœuds dans la portée d'un émetteur d'une session en cours sont interdits d'émettre, peut gaspiller la bande passante du réseau. D'autres problèmes tels que les pertes de paquets, l'atténuation de signal, etc., existent aussi dans les réseaux ad hoc à cause de l'interface sans-fil.

Nouveau besoin : parce que les solutions traditionnelles exigent souvent un échange fiable de messages, elles sont souvent non adaptées aux réseaux ad hoc. Les MANETs ont besoin de mécanismes tolèrant aux fautes et ayant un faible surcoût.

Dans la présente thèse, nous traitons en priorité des problèmes de sécurité causés par les nœuds compromis et l'impact de mécanismes de sécurité sur la performance du routage ad hoc. Nous prenons aussi des problèmes générés par la mobilité, l'égoïsme et la manque d'organisation en considération. Les problèmes causés par l'interface radio sont laissées pour les travaux futurs.

Motivations

Afin d'étudier systématiquement la sécurité du routage ad hoc, nous devons d'abord avoir une vue globale sur ses vulnérabilités. Actuellement, il y a déjà quelques travaux existants qui ont classifié les attaques des réseaux ad hoc, et beaucoup d'autres travaux ont étudié les attaques contre certain(s) protocole(s) de routage spécifique(s). Quant à nous, nous pensons qu'il est nécessaire de trouver une méthode d'analyse systématique capable d'analyser les vulnérabilités du routage ad hoc basées sur une vue générique des protocoles de routage ad hoc. D'ailleurs, nous devons aussi déterminer les vulnérabilités que nous allons traiter dans cette dissertation.
Et puis, après avoir analysé les vulnérabilités, nous devons ensuite étudier les solutions proposées pour la sécurité du routage ad hoc. Ayant constaté que beaucoup de

protocoles sécurisés et de mécanismes de sécurité pour les protocoles existants ont été suggérés, nous examinons particulièrement les mécanismes de sécurité fréquemment utilisés. Par exemple, le *watchdog* où les nœuds observent les comportements de leurs voisins afin d'identifier des attaquants.

Avec le *watchdog*, théoriquement toutes les opérations de tous les nœuds sont susceptibles d'être prises en compte dans la détection des attaquants. Il est donc important de ne pas se tromper dans l'authentification[1] des attaquants. Or, les protocoles existants utilisant *watchdog* ne résolvent pas ce problème pour une raison simple : il serait trop coûteux de tout authentifier. Nous voulons résoudre ce problème dans cette thèse, tout en améliorant la consommation de stockage dans le *watchdog*.

En outre, certaines études ont montré que, parce que les protocoles de routage réactifs génèrent moins de trafic de contrôle et peuvent gérer la mobilité d'une façon plus efficace que les protocoles de routage proactifs, ils sont mieux situés pour réaliser le routage ad hoc. Ainsi, nous commençons par étudier ces protocoles. Nous considérons que la cryptographie à elle seule n'est pas suffisante pour lutter contre beaucoup de problèmes de sécurité causés notamment par les nœuds compromis. Par conséquent, nous utilisons un mécanisme supplémentaire, en l'occurrence un système de réputation, pour isoler les nœuds compromis du routage.

Finalement, nous étudions aussi dans le routage proactif des MANETs, parce qu'il est très utile pour des scénarios et des applications qui ont besoin d'un court délai de routage. Le défi ici est de trouver des méthodes qui peuvent limiter la charge du réseau tout en sécurisant la topologie. Or, nous constatons que certains mécanismes de sécurité récemment proposés pour les protocoles proactifs peuvent dégrader la performance du routage ad hoc. Par conséquent, nous voulons alléger certains de ces protocoles sans pour autant baisser leur niveau de sécurité.

Contributions

Les vulnérabilités des réseaux ad hoc ont d'abord été analysées par une classification des attaques trouvées dans la littérature. Pour cela, nous avons utilisé le modèle de "l'arbre d'attaques" qui peut classifier les attaques en fonction de leur(s) objectif(s). Un arbre d'attaques est composé d'un objectif d'attaque commun (la racine de l'arbre), quelques sous-objectifs d'attaque (les branches de l'arbre) et finalement des mécanismes d'attaques (les feuilles de l'arbre). Cette méthode d'analyse présente deux avantages : premièrement, si nous voulons contrer un objectif d'attaque donné, il suffit de contrer toutes les attaques listées sous le sous-arbre de cet objectif; deuxièmement, pour connaître les vulnérabilités d'un protocole de routage, il est suffisant d'instancier l'arbre par ce protocole.

Ensuite, nous avons proposé un schéma de *watchdog* sécurisé que l'on l'appelle SWAN pour "Secured Watchdog for Ad hoc Networks". Ce mécanisme garantit l'authentification dans la supervision de *watchdog* en utilisant un schéma d'authentification sur la diffusion des messages. De plus, il fournit à *watchdog* un schéma efficace de

[1] Ici, l'authentification est utilisée pour associer les comportements à leurs auteurs origines.

gestion de stockage sans pour autant diminuer l'efficacité de détection de mauvais comportements du dernier. Dans SWAN, chacun des nœuds doit posséder une adresse temporaire basée sur une chaîne de hachage. Ceci permet d'une manière très simple et peu coûteuse de garantir l'authentification des messages de contrôle et de données auprès des nœuds observant. Le coût du SWAN au niveau de stockage et de calcul a aussi été étudié.
Par la suite, un protocole de routage sécurisé intégrant un modèle de confiance a été proposé et nommé TRP pour "Trust-based Routing Protocol". Ce protocole est basé sur DSR [JMH04] (pour "Dynamic Source Routing protocol") qui est un protocole de routage réactif. Le but principal de TRP est d'exclure les nœuds malicieux du processus de routage. De plus, la particularité de TRP, par rapport aux autres protocoles similaires comme CORE [MM02] et CONFIDANT [BB02b] qui utilisent eux aussi un modèle de confiance, est qu'il permet d'échanger des valeurs de confiance dans des messages de contrôle de routage tout en évitant les attaques de type *blackmail*[2]. Cette particularité peut contribuer à réduire la charge du réseau causée par les échanges de valeurs de confiance. De plus, le coût de la sécurisation d'échange de confiance est diminué parce qu'il est désormais possible de protéger à la fois les messages de contrôle de routage et les échanges de confiance.
Par rapport aux protocoles de routage réactifs, les protocoles proactifs semblent plus difficiles à sécuriser car leur quantité d'informations à sécuriser est plus importante. Cependant, ils ont un principal atout qui est que les nœuds connaissent en permanence la topologie du réseau entier grâce à l'échange permanent de messages de contrôle entre les nœuds. Nous avons choisi de sécuriser le protocole proactif OLSR (pour "Optimized Link State Routing protocole") [CJ03]. OLSR contient une amélioration importante par rapport aux protocoles de type *état de lien* traditionnels qui est l'utilisation de MPR (pour "MultiPoint Relay") (voir page xii pour plus de détails). Nous développons dans cette thèse deux mécanismes à faible coût, respectivement HPLS pour "Hash Proved Link State" et TCSec pour "Securing TC", afin d'empêcher les informations de routage forgées d'être acceptées par les nœuds bienveillants d'un réseau utilisant OLSR. L'idée principale de HPLS et de TCSec est l'utilisation d'informations supplémentaires (redondantes) afin de vérifier la validité des informations de routage.
Dans le reste de ce résumé, on va d'abord présenter les notations. Ensuite, nous allons développer un état de l'art analysant des solutions existantes de la sécurisation du routage ad hoc. Par la suite, nous proposons nos propres solutions dont SWAN, TRP, HPLS et TCSec. Finalement, le résumé termine avec une conclusion qui propose des considérations pour concevoir un nouveau protocole de routage ad hoc sécurisé dès le départ, et quelques perspectives dégagées par les travaux de cette thèse.

[2]Les attaques de type "blackmail" consistent à faire baisser les réputations des nœuds bienveillants par l'annonce de mauvaises recommandations contre ces nœuds.

Notation

Dans cette section, nous listons, dans l'ordre de leur apparition, les notations qui sont utilisées dans le présent résumé.

Notation	**Signification**
A, B	nœuds
S	nœud source
D	nœud destination
X	nœud malicieux ou égoïste
M	message/paquet
M_{Fix}	partie fixe du message M
M_{Var}	partie variable du message M
a	valeur
key	clé
$h_{key}(a)$	résultat de hachage de a en utilisant la clé key
h(a)	résultat de hachage de a sans clé
α, β	paramètres du modèle de confiance de TRP
HC	élément de hachage utilisé dans HPLS
i	intervalle

Routage dans les réseaux ad hoc

Dans le début des années soixante-dix, les réseaux ad hoc ont été premièrement inventés et étudiés pour les usages militaires. Depuis, leurs applications ont été largement étendues. Aujourd'hui, plusieurs standards ont été définis, comme par exemple 802.11 (aussi appelé Wi-Fi pour "Wireless Fidelity") au mode sans infrastructure, Bluetooth et HiperLAN.
Par ailleurs, beaucoup de protocoles de routage ont aussi été définis par des chercheurs. Ils peuvent être classifiés dans trois catégories :

réactif les nœuds échangent les informations de routage seulement quand il y a un besoin de découverte de route.

proactif les nœuds échangent entre eux des informations de routage en permanence afin que toutes les routes soient disponibles à tout moment.

hybride un mélange des deux premiers types de protocole.

Dans la suite, nous allons introduire brièvement les deux protocoles de routage ad hoc, respectivement DSR (réactif) et OLSR (proactif), que nous allons sécuriser dans cette thèse.

DSR (Dynamic Source Routing protocol)

DSR [JMH04] est un protocole de routage ad hoc réactif utilisant l'algorithme *routage par la source*. Trois principaux types de message de contrôle de routage sont définis dans DSR : RREQ pour "Route REQuest", RREP pour "Route REPly", et RERR pour "Route ERRor". Pour qu'un nœud source S envoie un paquet à un nœud destinataire D, S cherche d'abord dans son *cache de routes* s'il y a une route disponible pour D. Si oui, S envoie le paquet en utilisant la route trouvée. Sinon, S diffuse un paquet RREQ pour chercher une route vers D. Chaque nœud qui reçoit pour la première fois le RREQ et n'ayant pas de route vers D doit le rediffuser en rajoutant son identité dans la *route de source* du paquet. Par contre, en recevant le RREQ, D ou un autre nœud ayant une route vers D peut renvoyer un paquet RREP vers S déclarant la route trouvée (en utilisant la route cumulée dans la *route de source* du RREQ). Ainsi, S reçoit une route vers D en recevant chaque RREP. Ensuite, S peut utiliser les routes ainsi reçues pour envoyer son paquet. De plus, S va aussi sauvegarder ces routes dans son *cache de routes*. Si une route n'est plus valide lorsque qu'elle est utilisée pour une transmission de flux de données, un paquet RERR serait envoyé à S par le nœud en amont du lien cassé, pour que S change de route ou lance un nouveau processus de recherche de route.

OLSR (Optimized Link State Routing protocol)

OLSR [CJ03] est un protocole de routage ad hoc proactif basé sur l'algorithme *état de lien*. La technique clef de ce protocole est appelée MPR pour "MultiPoint Relay". Un MPR est un nœud choisi par son voisin pour transférer les messages de diffusion de ce dernier. Ainsi, au lieu que tous les voisins rediffusent les messages de diffusion, dans OLSR il n'y a que les MPRs qui vont les rediffuser. Cette amélioration peut largement contribuer à réduire la charge du réseau par rapport aux protocoles de routage du type *état de lien* traditionnels. Une deuxième amélioration aussi reliée à MPR est que le nombre de messages déclarant les états de lien est diminué, parce qu'il n'y a que les nœuds qui ont été choisis comme MPR les envoient. Une troisième amélioration est qu'un nœud MPR déclare seulement les liens avec ses sélecteurs.

Analyse de vulnérabilités du routage ad hoc

Les réseaux ad hoc sont exposés à un grand nombre de vulnérabilités, surtout au niveau routage. Étudier les vulnérabilités dans la couche réseau des MANETs peut nous permettre de reconnaître toutes les attaques à éviter, afin d'établir un environnement sécurisé qui satisfait les besoins de sécurité de chacune des applications de MANETs.
Il y en a déjà certains travaux existants. Par exemple, le travail dans [HJP02]

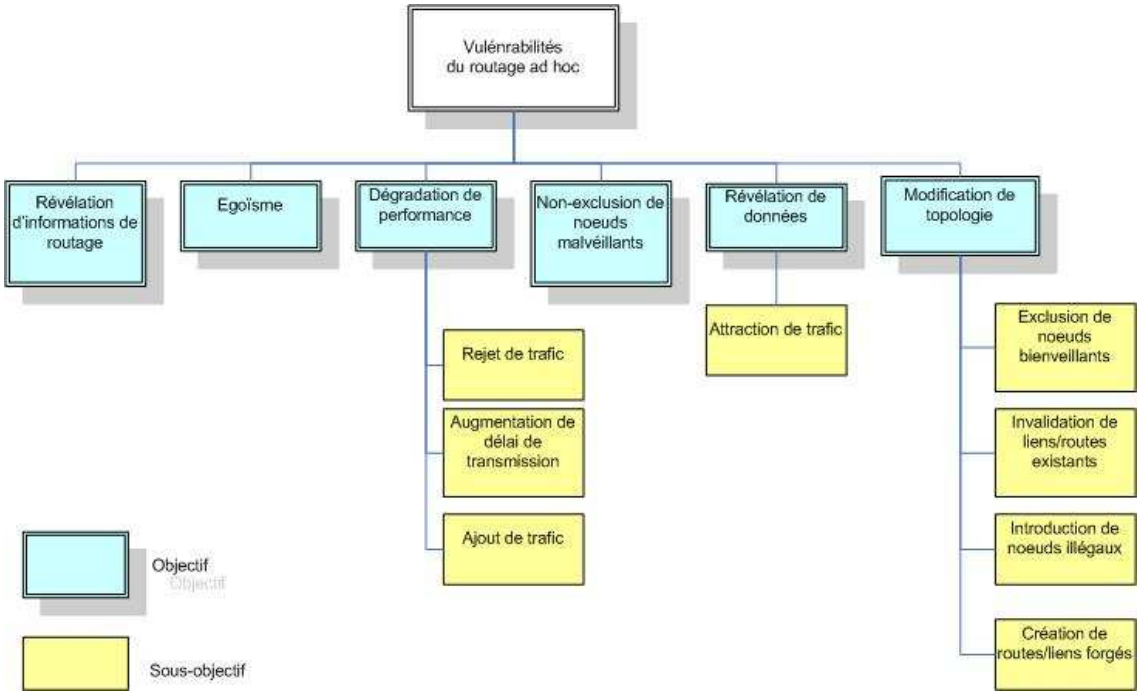


Figure 1: Arbre d'attaques (objectifs)

classifie les attaques dans les réseaux ad hoc en utilisant deux paramètres : le nombre d'attaquants coopérants internes (compromis) et le nombre total d'attaquants coopérants. Cette classification peut montrer le pouvoir des attaquants en fonction de leur nombre et de leur état de coopération.
Plus proche de notre modèle, un arbre d'attaques a été présenté dans [MM04]. Ce travail distingue d'abord les attaques passives et les attaques actives. Ensuite, les attaques actives sont classifiées en fonction de leur niveau d'action par rapport aux sept couches du modèle ISO.

Toutefois, nous constatons que les classifications existantes n'adressent pas toutes les vulnérabilités du routage ad hoc. De plus, leur modélisation n'est pas orientée suivant les objectifs de l'attaquant. Donc, nous réalisons une classification plus complète basée sur une vue générique des protocoles de routage ad hoc qui inclut le plus possible de vulnérabilités (à noter que nous ne travaillons que sur le niveau routage).
Un arbre d'attaque est de ce fait construit comme montré dans la figure ci-dessus. À noté qu'en raison de manque d'espace et à cause de la complexité de l'arbre, ici nous ne montrons pas l'arbre complet, mais seulement les branches de l'arbre (des objectifs d'attaques). Les lecteurs intéressés sont invités à lire le chapitre 2 de la thèse pour connaître les mécanismes d'attaques sous chaque branche.
Les objectifs principaux des comportements malicieux que nous avons constatés sont : la révélation d'informations de routage, la révélation de données, l'égoïsme, la dégradation de performance, la modification de topologie, et la non-exclusion de nœuds malicieux. Sous la branche "dégradation de performance", les trois sous-

objectifs sont les suivants : le rejet de trafic, l'ajout de trafic et l'augmentation de délai de transmission. Et puis, sous la branche "modification de topologie", il existe quatre sous-objectifs : l'exclusion de nœuds bienveillants, l'ajout de nœuds illégaux, l'invalidation de routes/liens existants, et la création de routes/liens forgés.

Cet arbre peut être instancié pour un protocole de routage existant ou nouveau, afin de montrer les vulnérabilités du protocole. Par exemple, la figure dans la page xv montre l'arbre d'attaques du protocole DSR. En analysant ces arbres, les attaques importantes à contrer sous chaque objectif peuvent être mises en lumière. Ainsi, la sûreté d'un version sécurisée de DSR ou OLSR. De plus, nous montrons aussi dans ces arbres les attaques considérées ou à ignorer dans cette thèse.

État de l'art de la sécurité du routage ad hoc

Beaucoup de travaux et d'efforts ont été consentis pendant ces dernières années pour la sécurité des réseaux ad hoc. Ces travaux peuvent être classifiés dans les trois catégories suivantes : la sécurisation du routage, la gestion des clés et le renforcement de la coopération.

Gestion de clés

Les solutions traditionnelles de gestion de clés ont dû trouver leur adaptation vis-à-vis des réseaux ad hoc, parfois en utilisant de nouvelles technologies. Par exemple, afin de pouvoir utiliser PKI (pour "Public Key Infrastructure") dans MANET, certains travaux [ZH99, ZSvR02, KZL+01, LL00, Sho00] ont employé la cryptographie à seuil ; le PGP (pour "Pretty Great Privacy") a été entièrement distribué pour les MANETs par [HBC01] en utilisant un cache de routes par nœud ; afin de supprimer totalement la dépendance aux serveurs de certificats (en anglais "Certificate Authority"), la cryptographie basée sur l'identité [BF01] et l'adresse basée sur cryptographie [MC02] ont été proposées.

En outre, pour établir une clé symétrique à l'échelle d'un réseau, une variante de Diffie-Hellman appelée "hybercube protocol" est décrite dans [AG00] ; et les grands réseaux hiérarchisés peuvent adopter la solution proposée dans [BHBR01].

Pour avoir des clés symétriques chacune partagée par un ensemble de nœuds, des techniques comme la pré-distribution de clés [CPS03, Cha04, EG02], le "Resurrecting duckling" [SA99, Sta01] et l'identification démonstrative [BSSW02] sont des solutions potentielles.

De plus, il y a aussi d'autres techniques sur l'utilisation de clés comme chaîne de hachage [Lam81], TESLA et μTESLA [PCSJ01, PCJS00], et IDHC [Mic04] ; et finalement l'établissement d'association de sécurité peut aussi être facilité par le routage [BEGA02] ou par la mobilité [CHB03].

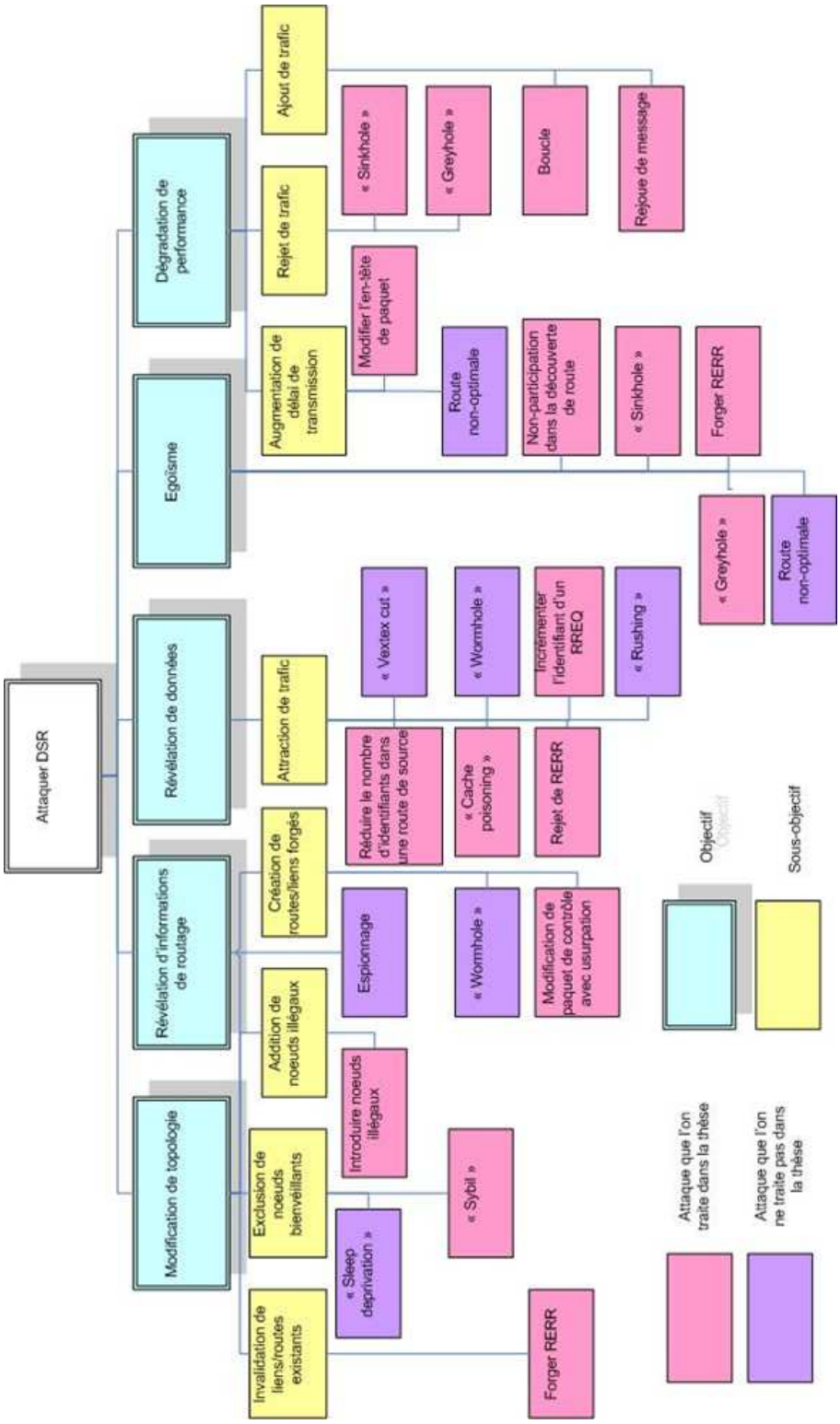


Figure 2: Arbre d'attaques de DSR

Sécurisation du routage

Pour sécuriser le routage, les objectifs suivants ont été identifiés : les routes peuvent être trouvées si elles existent (la disponibilité) ; une route en fonction doit au moins exister (l'exactitude) ; une route en fonction ne contient pas d'attaquant ou ne contient que des attaquants tolérables (la sûreté) ; les mécanismes de sécurité pour le routage ne doivent pas être trop lourds (l'efficacité de ressources) ; le routage sera stable en présence éventuel d'attaques non-contrées (la stabilité) ; et les données doivent être livrées correctement jusqu'à leurs destinations. Concrètement, ces exigences peuvent être ramenées à l'authentification et l'identification des nœuds, l'intégrité et l'authenticité des informations de routage, et l'intégrité, l'authentification et la confidentialité des données.
Les propositions représentatives des protocoles de routage proactif sécurisés sont par exemple [PH03, RACM04, HJP02], et celles pour le routage réactif sont par exemple [PZ03, PH02, HPJ02, ZA02, SDL+02, CY02, YNK02]. Il est généralement considéré que les protocoles du routage réactifs sont moins lourds en terme d'échange de messages que leurs homologues proactifs, et donc qu'ils sont plus faciles à sécuriser.

Renforcement de coopération

Concernant le renforcement de la coopération, il existe grosso modo trois sortes de solutions : les solutions basées sur un système de réputation, les solutions basées sur un modèle de "micro-payement", et les autres solutions.
Pour la première catégorie, CORE [MM02] et CONFIDANT [BB02b] sont deux propositions représentatives. Ces deux protocoles utilisent tous un module de type *watchdog*. Par ailleurs, CORE est validé par simulation et aussi par une modélisation en théorie de jeux ; CONFIDANT a deux versions dont la première suit le modèle de PGP en divisant les confiances en quatre niveaux et la deuxième suit le modèle de bayesian en séparant les niveaux de confiance sur les comportements et les niveaux de confiance sur les recommandations.
Les solutions basées sur le micro payement [ZCY03, BH01] suivent le principe suivant : les nœuds qui profitent du réseau (émetteurs et/ou récepteurs) payent les nœuds "fournisseurs de services" (nœuds intermédiaires). De cette façon, tout nœud doit servir les autres pour être servi lui-même.
Une autre solution [YML02] consiste à utiliser la cryptographie à seuil afin d'exclure collectivement les nœuds égoïstes. De plus, [JAA04] exige qu'un nœud intermédiaire (lui même aussi surveillé par ses propres voisins) change de route s'il juge que le nœud à qui il transfère des données est égoïste.

Discussion

Les solutions proposées pour la gestion de clés fournissent les primitives cryptographiques et les relations de confiance aux nœuds et aux autres mécanismes de sécurité du routage ad hoc. Quelques méthodes originales ont été utilisées pour adopter les solutions existantes aux MANETs. Néanmoins, les chercheurs sont toujours troublés par le problème ultime d'établir un schéma de clés sans infrastructure ni confiance a-priori.
Les solutions proposées pour le routage sécurisé sont essentiellement des utilisations des primitives cryptographiques dans les messages de routage ad hoc. Grâce à ces mécanismes, les nœuds peuvent être authentifiés, l'intégrité et la confidentialité des informations peuvent être assurées, et la plus part des attaques visant à exploiter les vulnérabilités du routage ad hoc peuvent être évitées. Toutefois, plusieurs de ces solutions impliquent un surcoût excessif de calcule et d'échange de données, qui est non-souhaitable pour les réseaux ad hoc à faible capacité.
D'autre part, le renforcement de la coopération est aussi capital pour les MANETs car les nœuds ad hoc ayant tendance naturellement à être égoïstes à cause de leur faible capacité. Une des solutions est d'employer un modèle de confiance pour quantifier le niveau de coopération de chacun des nœuds, et une des questions importantes à se poser est l'utilisation de recommandations qui ne sont pas toujours fiables dans ces modèles.
Dans la suite, nous traitons d'abord le problème d'authentification par les mécanismes de type *watchdog*. Ce problème est ignoré par les propositions existantes qui font l'hypothèse que les identités des nœuds soivent toujours unique et exact. Nous proposons une solution pour supprimer cette hypothèse et aussi pour diminuer la consommation de stockage.
Ensuite, nous réfléchissons sur la possibilité d'intégrer un système de confiance directement dans le routage. C'est-à-dire, les messages de confiance sont échangés durant les échanges d'informations de routage, et les deux types d'échanges sont sécurisés en même temps. De cette manière, nous pouvons éviter dans la plupart de cas les attaques de type *blackmail*.
Le dernier problème que l'on traite dans cette thèse est la sécurité dans le protocole proactif OLSR. Nous proposons deux solutions légères qui renforcent la sécurité d'OLSR.

SWAN (Secured Watchdog for Ad hoc Networks)

Nous proposons un schéma d'authentification pour le mécanisme *watchdog* afin de garantir que les identités des nœuds ad hoc soient unique et exact dans des protocoles tels que CORE et CONFIDANT. Le processus de *watchdog* est illustré par la figure 3 où A peut comparer les paquets 1 et 2 pour rechercher les éventuelles anomalies selon

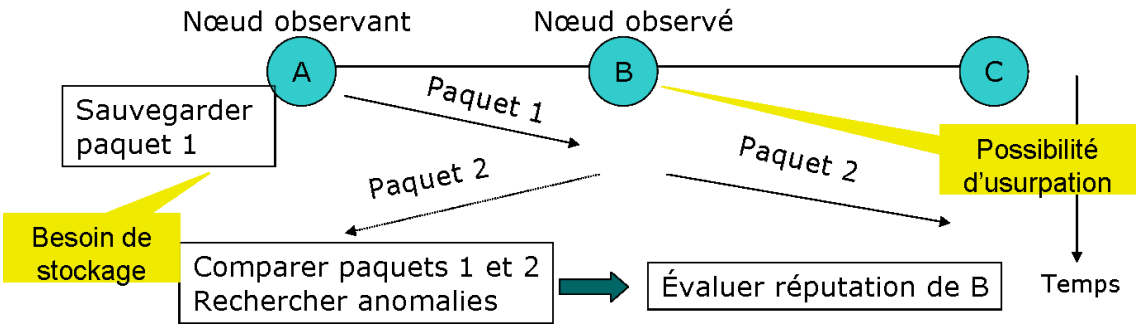


Figure 3: Mécanisme de watchdog

lesquelles A peut évaluer sa réputation sur B. Par conséquent, il est nécessaire que l'identité de B soit unspoofable et unforgeable, et que le besoin de stockage (pour stocker les paquets à observer) du nœud observant A soit non-excessif.
Le mécanisme SWAN pour "Secured Watchdog for Ad hoc Networks" regroupe les caractéristiques des deux mécanismes existants : SUCV [MC02] et TESLA [PCSJ01]. Ses principaux avantages sont, premièrement, que l'on peut authentifier un grand nombre de paquets avec un faible coût, deuxièmement, il n'est pas nécessaire d'avoir un serveur. Cependant, notre schéma a un désavantage important : il tolère des adresses fictives.
Figure 4 illustre un exemple d'utilisation de SWAN. Le temps du réseau est divisé en plusieurs intervalles. Chaque nœud possède une chaîne de hachage et va utiliser un élément de la chaîne comme clé pour authentifier ses messages envoyés durant un intervalle. L'élément sera ensuite révélé durant le prochain intervalle, et le nœud observant peut ainsi vérifier l'authentification de l'émetteur et l'intégrité du paquet. L'économie de stockage est réalisée par la distinction des parties fixe M_{Fix} et variable M_{Var} d'un paquet M et l'ajout d'un champ supplémentaire $h_{key}(h(M_{Fix})|M_{Var})$ au paquet, où key est la clé utilisée pour l'authentification.

TRP (Trust-based Routing Protocol)

Le protocole de routage basé sur confiance (TRP pour “Trust-based Routing Protocol”) [XLB04] est notre proposition de protocole sécurisé basé sur DSR. Il sécurise les deux phases du routage : la découverte de topologie et l'acheminement des données, et détecte les deux sortes de comportements anormaux : actes égoïstes et actes malicieux. La fxigure 5 nous montre les modules de TRP : un watchdog est utilisé pour alimenter les échanges de réputations qui sont intégrées dans les messages de routage afin de mieux réaliser les choix de route.
Figure 6 nous explique les trois types de confiance utilisés dans TRP, respectivement Confiance Directe (CD), Confiance Indirecte (CI) et Confiance sur Route (CR). Un système de réputation alimenté par des “observations sur routes” est adopté pour aider aux nœuds sources dans leurs choix de routes. Dans TRP, les observations

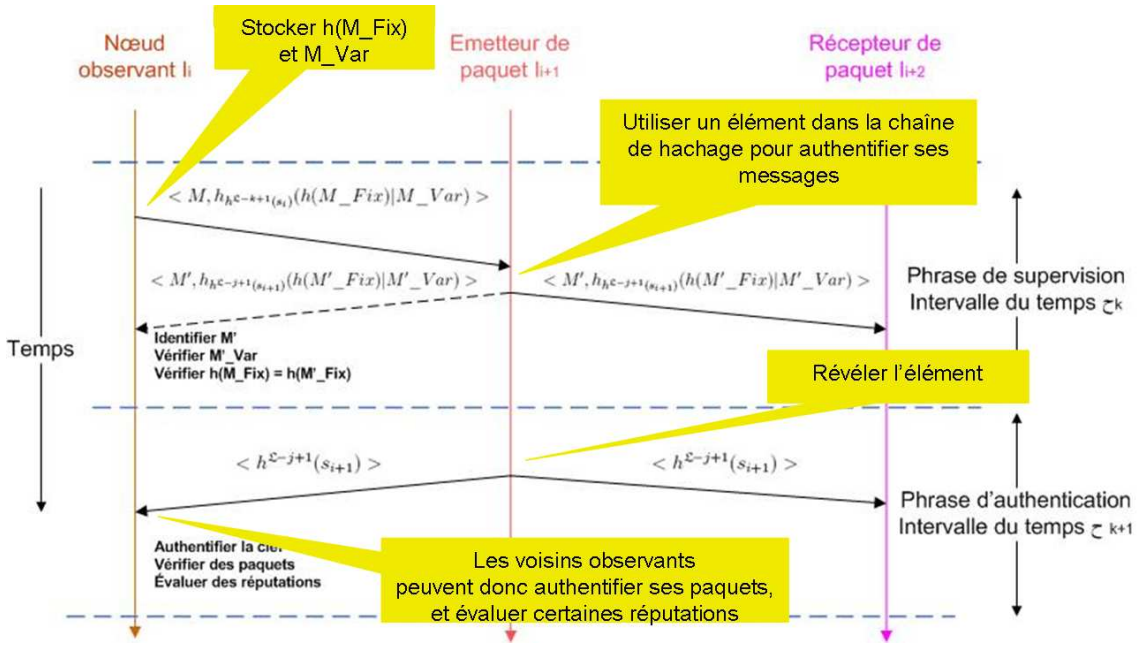


Figure 4: Exemple de SWAN

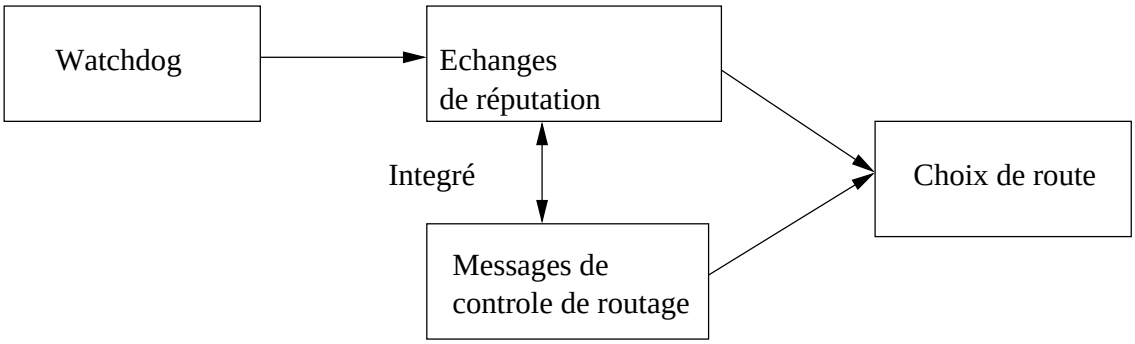


Figure 5: Modules de TRP

directes sont toujours prioritaires par rapport aux recommandations qui sont utilisées seulement en absence d'observation directe. De plus, les recommandations ne doivent pas être utilisées directement non plus : elles devraient d'abord passer par un calcul de "confiance indirecte". Finalement, les recommandations ne sont jamais enregistrées pour un temps long, car elles ne sont utilisables que pour le choix de la route actuelle. Toutes ces mesures y compris la protection de l'intégrité des paquets nous permettent d'éviter les attaques de type "blackmail". Afin de choisir une bonne route, un nœud source prend normalement une route ayant une valeur de CR élevée. Le protocole suppose qu'il existe entre chaque couple d'émetteur et récepteur au moins une route qui ne contient pas d'attaquant, et l'objectif du TRP est bien de la trouver pour acheminer des données. Côté cryptographie, on fait l'hypothèse qu'une clé est pré-partagée entre chaque source et sa destination. Le schéma de routage est donc similaire à SRP [PH02]. De plus, au fur et à mesure durant la propagation des RREQ, des valeurs de confiance se cumulent. Finalement ces valeurs vont être

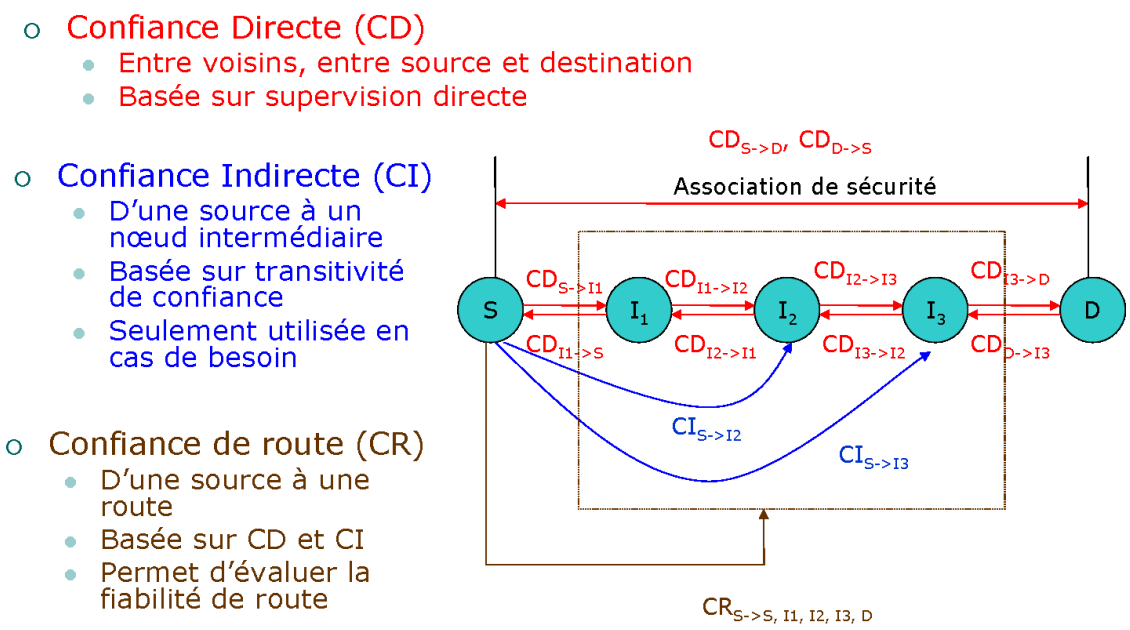


Figure 6: Relations de confiance dans le protocole TRP

renvoyées à la source pour l'aider à choisir une route sécurisée. La clé partagée sert à protéger l'intégrité de toutes les informations transmises dans les RREP. Cette procédure est expliquée dans figure 7.
Une différence importante entre TRP et CORE ou CONFIDANT est que nos échanges de valeurs de confiance sont intégrés dans les messages de contrôle de routage. Cette mesure permet de réduire la charge du réseau causée par ces échanges, et en plus de protéger ces échanges en même temps qu'on protège les messages de routage.

TRP$^+$

TRP$^+$ est une amélioration de TRP. TRP n'utilise que les relations de confiance dans un seul sens sur chaque route, alors que TRP$^+$ utilise les valeurs de confiances dans les deux sens. Sachant que le sens de vérification rajouté est aussi le sens des données à acheminer, cette modification fait que TRP$^+$ est plus efficace en repérant et en excluant les nœuds malicieux ou égoïstes sur une route. La particularité de TRP$^+$ par rapport TRP est illustré dans la figure 8, et nous pouvons trouver la comparaison de performance entre TRP et TRP$^+$ dans la figure 9.

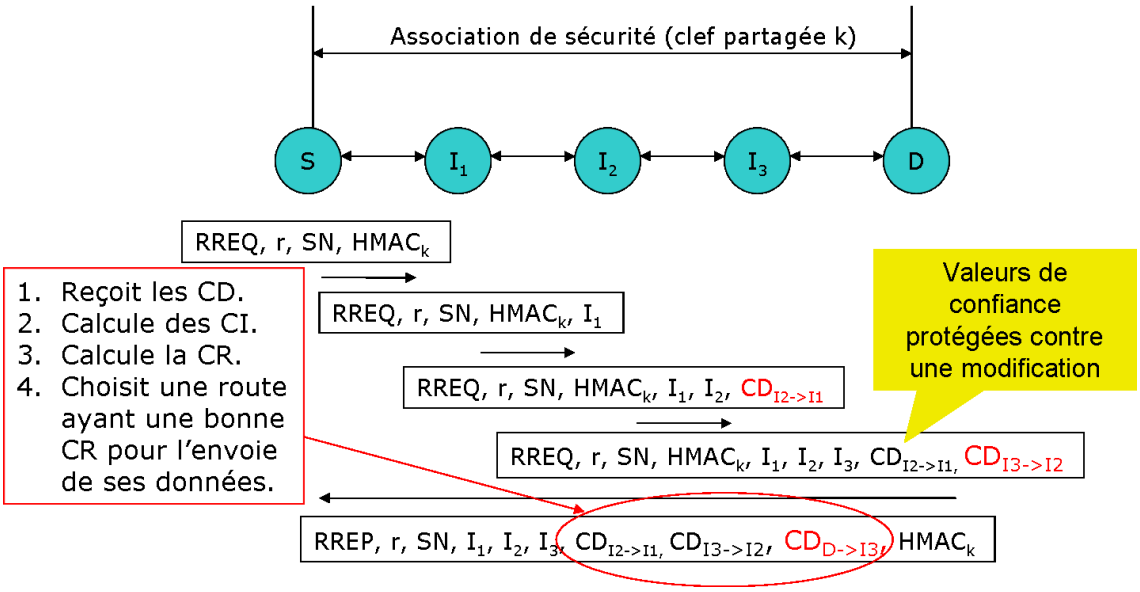


Figure 7: Illustration du protocole TRP

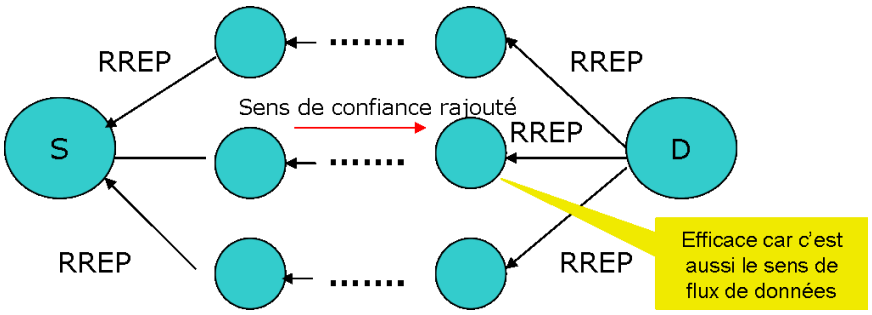


Figure 8: TRP+ par rapport au TRP

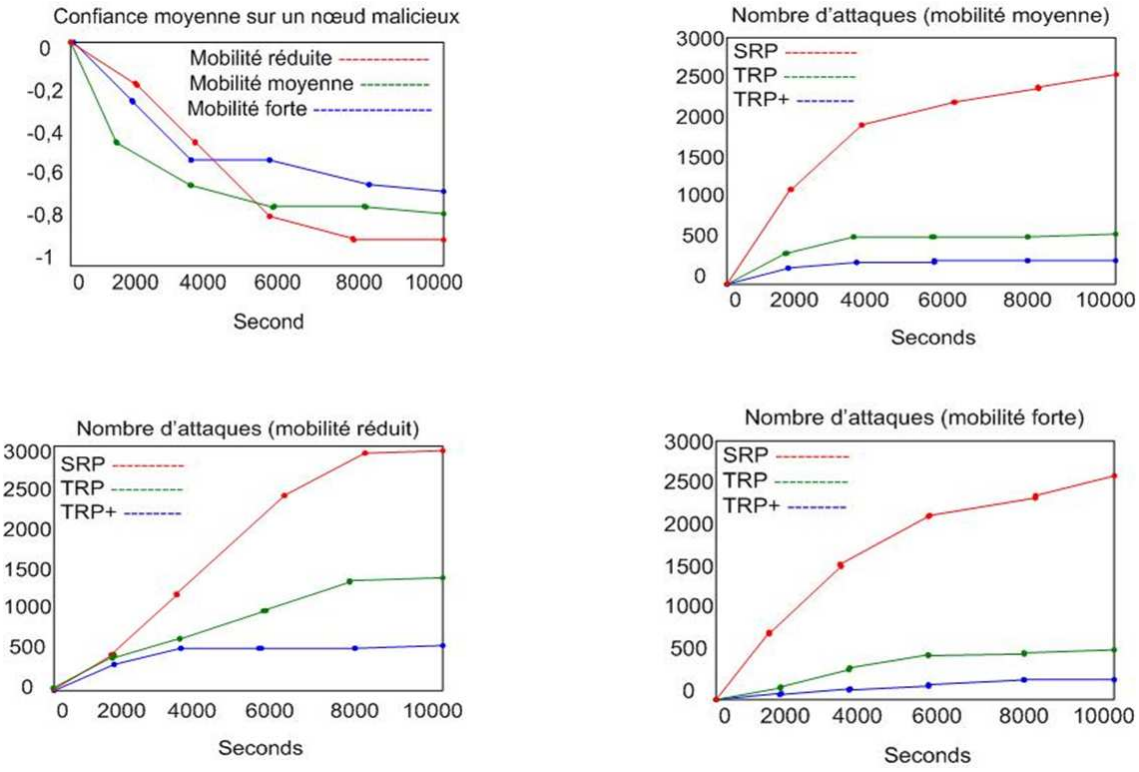


Figure 9: Résultats de simulations de TRP

TRPS (TRP avec SWAN)

Comme SWAN est applicable au watchdog, il est aussi applicable sur TRP. Nous l'avons donc intégré dans TRP, et avons ainsi créé un nouveau protocole TRPS pour "TRP avec SWAN". Nos simulations montrent que TRPS réduit le besoin de stockage de TRP et est plus sécurisé contre les attaques d'usurpation d'identité.

Résultats de simulation

Les simulations ont été effectuées sous NS-2 en utilisant la librairie d'OpenSSL pour la cryptographie. L'implémentation de TRP a été réalisée à partir de l'implémentation de DSR.
Le réseau a une taille de 700m*700m et comprend 25 nœuds dont un nœud malveillant. Le comportement implémenté pour ce dernier est le suivant : il modifie les données ou les en-têtes quand il les transfère, et, de plus, il n'envoie jamais de message RERR.
Trois scénarios de mobilité basés sur le modèle "random way-point" ont été testés :

- faible mobilité - temps d'arrêt de 100 secondes et vitesse maximale de 2 m/s,
- mobilité moyenne - temps d'arrêt de 20 secondes et vitesse maximale de 5 m/s,

- mobilité forte - temps d'arrêt de 5 secondes et vitesse maximale de 20 ms.

L'application considérée est FTP et les flux sont de type CBR. Un scénario de trafic est généré aléatoirement, les tests étant effectués avec 22 connexions et chaque connexion ayant au maximum 2000 paquets à envoyer et un taux d'envoi à 2pqts/s. La taille du tampon "promiscuous" est de 30 entrées. Enfin, en ce qui concerne le calcul des valeurs de confiance, nous avons considéré les deux paramètres de TRP $\alpha = 0.75$ et $\beta = 10$.
Les simulations ont été réalisées pour TRP, TRP$^+$ ainsi que pour TRPS.
Les résultats de simulation nous montrent que :

- TRPS peut sécuriser le watchdog : en simulant certains attaquants qui attaquent avec l'usurpation d'identité, on constate que TRPS peut éviter jusqu'à un certain degré que les mauvaises réputations soient affectées au nœuds bienveillants.

- dans TRP la moyenne des valeurs de confiance directe sur l'attaquant diminue effectivement avec le temps, quel que soit le type de mobilité ; et puis, pour chaque scénario de mobilité, le nombre d'attaques réussies se stabilise lorsque TRP est implémenté, les résultats étant légèrement améliorés dans le cas de TRP$^+$; en outre, comme on pouvait le prévoir, plus la mobilité est forte, plus TRP est efficace : un changement fréquent de topologie du réseau augmente la probabilité qu'un nœud soit dans le voisinage de l'attaquant et puisse l'observer et ainsi le détecter.

- côté performance, la longueur moyenne des routes n'augmente pas ; en terme de communication, aucun nouveau message n'est ajouté, seule la taille des messages RREQ et RREP augmente légèrement du fait de l'ajout des entêtes ; cependant la charge de routage (le nombre total de paquets de routage émis pendant la simulation) augmente de manière non négligeable par rapport à DSR (cela provient principalement du fait que certaines optimisations de DSR ont été supprimées dans TRP, et il devient nécessaire de rafraîchir le cache plus fréquemment) ; concernant le délai de bout-en-bout (c'est-à-dire le temps écoulé entre le moment où le message est créé au niveau applicatif et le moment où il est délivré à la destination), nous n'avons pas observé de différence significative avec DSR ; finalement, les calculs réalisés pour assurer l'intégrité des messages de routage et évaluer les valeurs de confiance sont relativement simples, et le délai induit par leur coût est négligeable.

Sécurisation d'OLSR

Pour sécuriser le protocole OLSR, quelques solutions ont été proposées [WHiKlS05, WlSiK05, HHF05, ACL$^+$05], notamment une approche appelée ADVSIG (pour "ADVanced SIGnature") [RACM04]. Néanmoins, à cause de la quantité importante

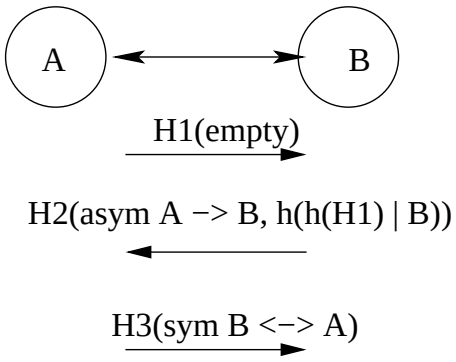


Figure 10: Exemple d'ADVSIG$^+$

d'informations à sécuriser, ces solutions sont soit incomplètes puisqu'elles traitent seulement une partie des problèmes de sécurité, soit trop lourdes en terme de le surcharge de messages de contrôle et/ou de calcul cryptographique.

ADVSIG$^+$

Il existe une faille de sécurité dans l'ADVSIG : ADVSIG peut confirmer un faux état de lien de type "asymétrique". Ainsi, un attaquant peut avoir la chance de créer un lien symétrique fictif alors que ce lien est seulement asymétrique. Cette attaque peut influencer temporairement les choix des nœuds MPR donc certains tableaux de routage.
ADVSIG$^+$ est une variation d'ADVSIG que nous proposons pour améliorer la sécurité d'ADVSIG. La figure 10 nous montre l'idée d'ADVSIG$^+$. La différence par rapport à ADVSIG est que nous rajoutons un champ supplémentaire pour confirmer un état asymétrique, et que ce champ dépend du précédent message H1 et de l'identité de nœud qui envoie le message H2. Par conséquent, un attaquant n'ayant pas reçu le message H1 ne peut plus forger un message H2 comme dans ADVSIG.

HPLS (Hash Proved Link State)

Nous avons proposé une approche HPLS pour sécuriser OLSR d'une manière bien plus légère qu'ADVSIG. Pour cela, nous supposons que l'on connaît le nombre maximal de nœuds dans un réseau et le temps maximal de l'existence du réseau.
L'hypothèse de départ est que chaque nœud possède plusieurs chaînes de hachage (chaque chaîne représente un état tel que "asymétrique", "symétrique" ou "MPR" entre une paire de nœuds) qui sont en relation avec son identité. De plus, chaque élément dans ces chaînes doit pouvoir représenter et confirmer un état de lien entre un pair de nœuds dans un petit intervalle de temps.
Afin de lutter contre les liens forgés, ces éléments sont gardés secrets avant leur révélation, donc la connaissance des éléments peut être considérée comme une méth-

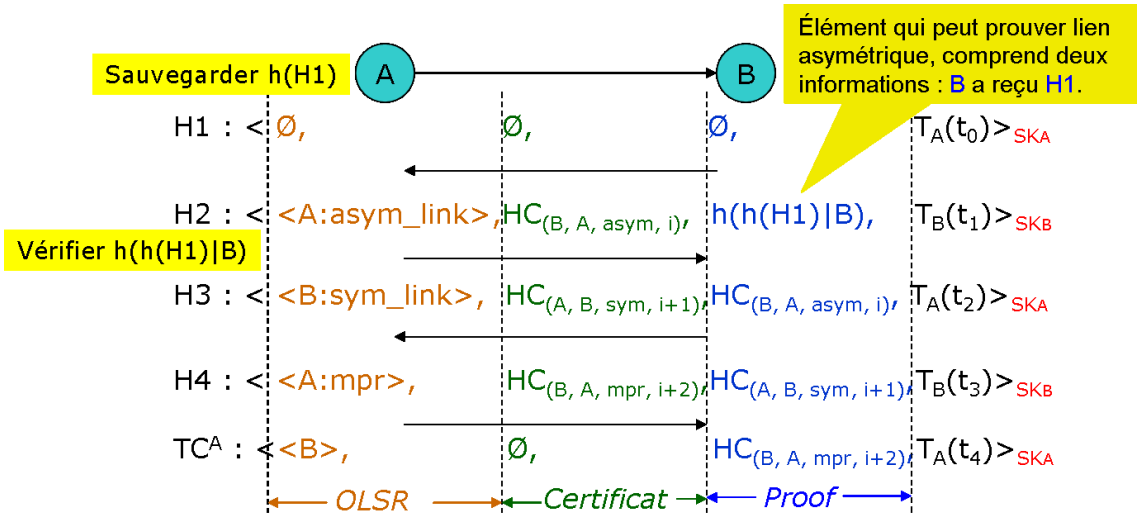


Figure 11: Idée basic de HPLS

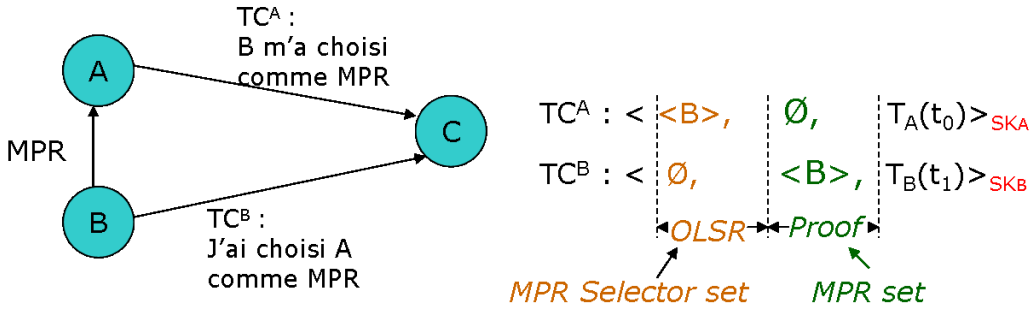


Figure 12: Principe de TCSec

ode pour l'authentification des liens.
La figure 11 nous montre quelques messages de type HELLO envoyés entre deux nœuds voisins A et B pour établir une relation MPR entre eux. Avec HPLS, nous souhaitons avoir moins de surcharge de calcul et de messages de contrôle qu'ADVSIG. Ceci est réalisé par l'utilisation d'éléments de hachage. Par exemple, un élément $HC_{(B,A,asym,i)}$ veut dire que le nœud A considère qu'il a un lien asymétrique avec B dans l'intervalle i. À la réception de chaque message, les *proofs* sont vérifiés afin d'authentifier les liens déclarés.
D'ailleurs, l'analyse en détail des nœuds multi-interfaces peut être trouvée dans la thèse, section 6.4.2.4.

TCSec (Topology Control Security)

Le principe de TCSec est illustré dans figure 12. Supposons qu'un nœud A est choisi par un autre nœud B comme MPR, alors non seulement on demande au nœud A de le déclarer comme initialement prévu par le protocole OLSR, mais on exige également

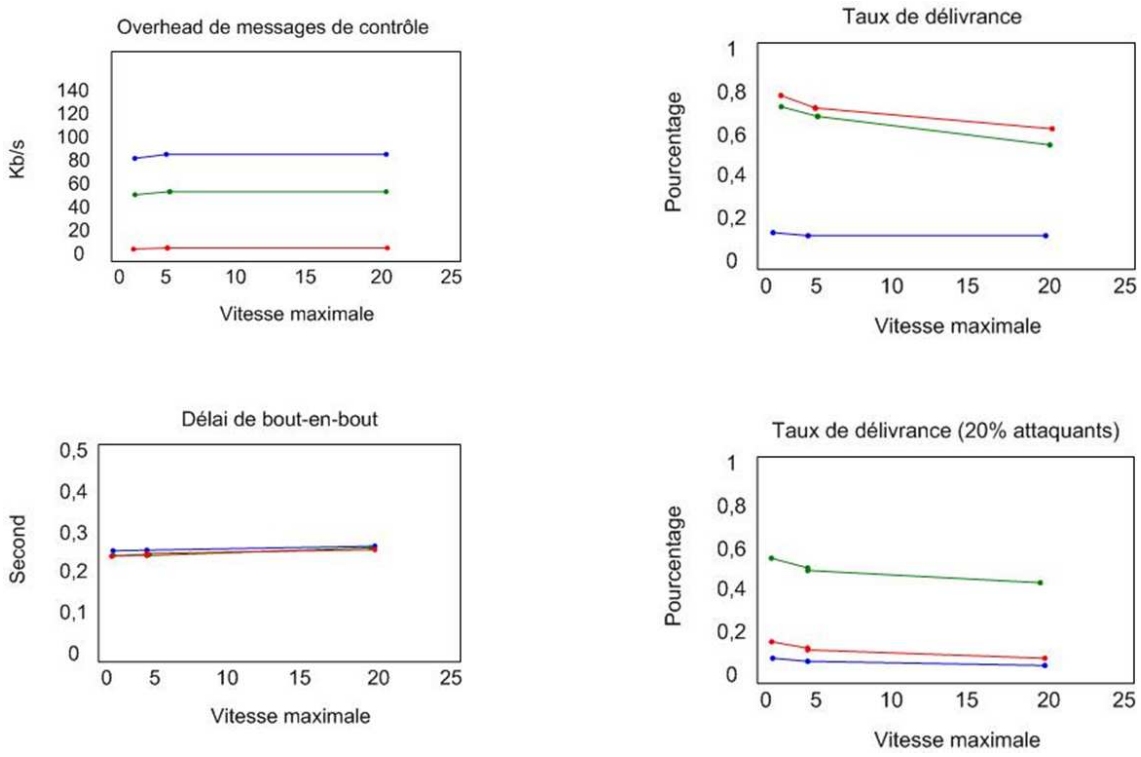


Figure 13: Résultats de simulations pour la sécurisation d'OLSR (ligne rouge: OLSR original, ligne bleu: ADVSIG, ligne verte: notre approche)

que le nœud B déclare dans son message TC qu'il a choisi A comme MPR. Ainsi, chaque autre nœud peut comparer les deux déclarations. S'il y a une incohérence entre elles, la relation de MPR ne sera pas considérée fiable. Cette mesure permet de sécuriser les messages TC lorsqu'il n'y a pas d'attaquants coopérants qui déclarent en commun une relation de MPR fictive.

Résultats de simulation

Les simulations ont été réalisées sous NS-2 et basées sur l'implémentation d'OLSR de l'Université de Murcia. La surface simulée est 300m*1500m (correspond à une région d'une autoroute). Les trois vitesses maximales simulées sont respectivement 2m/s, 5m/s et 20m/s. Le trafic est au maximum 20 CBR flux à 10 paquets par second et à 64 octets par paquet.

Il faut noter que nous avons pris soin d'insérer dans nos simulations le vrai temps nécessaire pour chaque opération cryptographique inclue dans nos mécanismes. Par exemple, pour préparer une signature à 320 bits de longueur, on a choisi un temps aléatoire entre 0.2ms et 150ms (afin de considérer un réseau hétérogène avec les nœuds de capacité différente) mais fixe pour chaque nœud. Cette mesure nous permet de voir le vrai effet du délai introduit par l'utilisation de cryptographie dans le routage des réseaux ad hoc, car une information de routage est validée et utilisée

seulement après le délai nécessaire de vérification, et un message peut être envoyé dans le réseau seulement après le délai nécessaire pour le signer.
Nous avons utiliser HPLS sur les messages HELLO et TCSec sur les messages TC. Nos mécanismes ont été comparés avec l'OLSR original sans mécanisme de sécurité et à une solution existante ADVSIG (pour "ADVanced SIGnature"). Les trois métriques mesurés dans nos simulations sont :

- surcharge causé par les messages de contrôle ; les simulations ont montré que, par rapport à l'OLSR original, nos mécanismes de sécurité ont introduit un overhead supplémentaire non-négligeable, mais pour avoir un même niveau de sécurité qu'ADVSIG, nous avons entre 30 à 35% de l'overhead de moins.

- taux de réussite de délivrance de données ; les résultats de simulation montrent que, par rapport à l'OLSR original, notre approche dégrade le taux de délivrance de 5 à 8%, cependant il peut largement améliorer la performance d'ADVSIG grâce au faible coût de nos opérations cryptographiques.

- moyen de délai de bout-en-bout pour les paquets de données ; notre approche a un moyen de délai de bout-en-bout un peu inférieur à celui d'ADVSIG et similaire à celui d'OLSR.

Les résultats de simulations sont montrés dans figure 13.

Conclusion générale

Les réseaux ad hoc présentent des challenges difficiles dans la sécurisation du routage. Nous devons non seulement éviter de nombreuses attaques causées par les attaquants externes et les nœuds compromis, mais aussi assurer que la dégradation de performance causée par les mécanismes de sécurité soit limitée.
Dans cette thèse, nous avons d'abord classifié, dans un arbre d'attaques, les menaces connues contre la couche réseau des MANETs. On distingue dans cette classification les objectifs d'attaques et les mécanismes d'attaques, et ceci va nous permettre de déterminer les attaques à contrer sous chaque objectif de sécurité. Deuxièmement, nous avons identifié certaines nouvelles vulnérabilités causées par l'utilisation des mécanismes de sécurité. Troisièmement, on a aussi instancié l'arbre d'attaques pour deux protocoles de routage, DSR et OLSR, afin de montrer leurs vulnérabilités. Finalement, nous avons aussi identifié, à l'aide de l'arbre, les attaques que nous traitons dans cette thèse.
Par la suite, un état de l'art des propositions existantes, de gestion de clés et de renforcement de la coopération, a été réalisé. Nous avons constaté à travers cet état de l'art que, d'une part, il y a des mécanismes non sécurisés souvent utilisés par les solutions de sécurité, comme par exemple, chaînes de hachage, *watchdog*, système de réputation, etc.; d'autre part, quelques protocoles de sécurité peuvent générer un

coût élevé, ce qui est non-souhaitable pour les MANETs qui ont une bande passante et/ou une puissance de calcul limitée.

Ensuite, on a suggéré un watchdog sécurisé appelé SWAN, dans lequel on combine SUCV et TESLA afin de développer un schéma d'authentification de message de diffusion. SWAN peut être utilisé pour réduire le besoin de stockage dans watchdog et pour empêcher les attaques d'usurpation qui peuvent mal affecter les systèmes de réputation. Notre analyse montre que SWAN est léger et robuste, donc remplit les objectifs de cette recherche. De plus, SWAN convient aux protocoles de routage de type "source routing", où le contenu des paquets est prévisible.

Nous avons aussi proposé le protocole TRP (Trust-based Routing Protocol). TRP est un protocole réactif basé sur l'algorithme de "source routing" et un système de réputation qui donne des mesures de confiance aux nœuds. On distingue deux phases de routage ad hoc : la phase de découverte de topologie et la phase de délivrance de données. Dans la première phase, nous utilisons un HMAC pour protéger les messages de contrôle de bout-en-bout, et un mécanisme de type watchdog pour superviser les attaques. Dans la deuxième phase, le watchdog est aussi utilisé pour superviser les attaques et les mauvais comportements commis sur les paquets de données. Le système de réputation basé sur watchdog va ensuite donner un niveau de confiance à chaque route. Ainsi, chaque nœud source aura la possibilité de choisir la route la plus sûre pour envoyer ses données. Grâce à ces mesures, les nœuds malicieux et les nœuds égoïstes vont être isolés du réseau.

Enfin, nous avons proposé deux mécanismes de sécurité pour le protocole de routage OLSR. Le premier mécanisme HPLS utilise des éléments de hachage pour authentifier les états de liens, et le deuxième mécanisme TCSec vérifie la cohérence des déclarations de relation de MPR. Toutes les deux approaches sont très légères et n'affectent pas la performance du réseau d'une manière significative.

Concevoir un nouveau protocole de routage ad hoc sécurisé dès le départ

Dans cette thèse, nous avons étudié les mécanismes de sécurité existants pour les protocoles de routage ad hoc, et nous avons aussi proposé quelques nouveaux mécanismes pour sécuriser respectivement le watchdog, le protocole de routage réactif DSR et le protocole de routage proactif OLSR. Et nous avons abordé le problème de performance causé par la sécurité.

Néanmoins, sécuriser un protocole de routage existant n'est peut-être pas la meilleure façon de sécuriser les MANETs. Il pourrait être plus sûr de désigner des nouveaux protocoles de routage sécurisés dès le départ.

Avec les renseignements obtenus au long de cette thèse, nous pensons que de tels protocoles doivent avoir les trois éléments de base suivant : la détection sécurisée de voisins, l'authenticité d'information de routage, et des mesures de sécurités contre les nœuds compromis.

Perspectives

Les travaux réalisés dans cette thèse nous a permi d'élaborer les perspectives suivantes pour le court-terme :

- le système de réputation utilisé par le TRP mérite quelques études plus approfondies ; une analyse formelle sera nécessaire pour démontrer théoriquement sa justesse et son efficacité ; d'autres simulations sont aussi à effectuer pour ajuster ses paramètres avec des scénarios typiques des réseaux ad hoc.
- pour les protocoles de sécurité proposés dans cette thèse, nous avons aussi besoin de valider leur propriétés de sécurité ; par exemple, une analyse formelle sera appréciée si elle peut prouver que l'attaque de type "link spoofing" ne peut pas avoir lieu dans HPLS et TCSec.
- en plus de SWAN, on peut rechercher de solutions pour les problèmes de nœuds cachés qui peuvent influencer de manière néfaste les résultats de supervision.

D'ailleurs nous pensons que les directions suivantes de recherches nécessitent de plus amples investigations dans un long terme :

- l'arbre d'attaques présenté dans l'analyse de vulnérabilités du routage ad hoc peut être enrichi chaque fois qu'il y a une nouvelle vulnérabilité retrouvée.
- nous pouvons fournir aux simulateurs, comme par exemple NS-2, GlomoSim, etc., la capacité de prendre en compte avec une simple configuration le délai de calcul généré par des opérations cryptographiques, et la facilité d'effectuer des simulations dans un environnent hostile.
- les menaces de sécurité ne sont pas limitées à la couche réseau ; les mécanismes de synchronisation, les protocoles de transport, les protocoles de contrôle d'accès, peuvent aussi être des cibles d'attaques ; donc nous devons aussi considérer les problèmes de sécurité sur les autres couches ; par exemple, les protocoles de contrôle d'accès au média sont aussi intéressants à étudier.

Abstract

Mobile Ad hoc Networks (MANETs) refer to mobile and wireless networks independent of any infrastructure. Instead of using designated routers to forward data as in traditional networks, in a MANET every node should participate in the routing. Some ad hoc scenarios are in a hostile environment. Moreover, due to numerous constraints such as the lack of infrastructure, the lack of a-priori trust relationship, resource-constrained nodes, mobility, etc., the ad hoc routing is vulnerable to numerous attacks.

Currently, a large number of ad hoc routing protocols such as Optimized Link State Routing protocol (OLSR) and Dynamic Source Routing protocol (DSR) have been proposed. However, few of them have seriously considered the security issues from scratch.

In this dissertation, we first present a classification of ad hoc routing vulnerabilities using the attack tree analysis model. The main characteristic of this work is that we distinguish objectives and mechanisms of attacks. This distinction can help security defenders to easily notice which attacks should be prevented under which security objectives, and which attacks are not required to be countered.

We then focus on our main research objective which is the proposition of new secure mechanisms for ad hoc routing protocols. We also pay attention to limit the performance degradation caused by security mechanisms.

We propose at first a Secure Watchdog for Ad hoc Networks (SWAN). It uses a broadcast message authentication scheme to ensure the authentication in supervision. In addition, it can also reduce the storage requirement of watchdog. The security and overhead analysis of SWAN show that the scheme is both robust and lightweight.

Besides, we propose a Trust-based Routing Protocol (TRP) which uses DSR as its underline routing algorithm. TRP employs a trust model and integrates the reputation exchanges into routing control messages. Moreover, SWAN can also be applied to TRP. The simulation results show that malicious nodes can be identified and isolated, if they commit attacks.

Then we study the security of the OLSR protocol and suggest two mechanisms to improve its security. One mechanism called Hash Proved Link State (HPLS) uses Hash values to prove the link relationships between nodes, and the other mechanism called Securing Topology Control (TCSec) uses coherence check to secure TC messages in OLSR. Simulations show that our solutions offer a good trade-off between the security and the routing performance.

Finally, we use the experience obtained in this thesis to provide some guidelines for the design of a new ad hoc routing protocol secured from scratch.

Contents

List of Figures

List of Tables

Notations

Time

t	time
t_i	the ith timestamp
T_Max	upper bound of the lifetime of the network
$T_A(t_i)$	timestamp at local time t_i of node A
$T0$	the network starting time
Δt	the duration of a time interval
τ	time interval
τ_i	the ith time interval
e	expiration time
c_j	the jth cycle in the network

Math

r	random integer
$[x/y]$	y integer divides x
s	random seed
s_A	random seed chosen by node A
$s_{A;j}$	random seed chosen by node A for the cycle c_j
s_i	random seed chosen by node n_i
$s_{i;j}$	random seed chosen by node n_i for the cycle c_j
${}^xP_y$	number of permutations of x elements taken y at a time
${}^xC_y$	number of combinations of x elements taken y at a time
$i\|j$	the catenation of i and j

Cyptography

- Server -

CA	CA server
T	off-line key server

- Key and certificate -

K	symmetric key
$K_{A,B}$	symmetric key shared by node A and node B
PK	public key
PK_A	the public key of node A
SK_A	the private key of node A
$cert_A =< IP_A, PK_A, t, e >_{SK_{CA}}$	the certificate of node A
$K_{I_{i\tau_i}}$	the TESLA key of node I_i at time interval τ_i
TIK_i	the TIK key at time t_i

- Hashing and hash chain -

l	the length (in bits) of hash values
$hash - i(a)$	the i-bit hash ouput of a
$h(a)$	the hash value of a
$h^j(a)$	value a hashed j times without key
$h_{key}(a)$	HMAC computed on a value a using the key key
h^m	the mth element in a Hash chain
$\mathfrak{L}$	the length of a Hash chain
HC_k	the kth Hash chain
$seed_k$	seed of the Hash chain HC_k
$HC_{k;m}$	the mth hash value of the Hash chain HC_k; it equals to $h^m(s_k)$

- Others -

ns	number of threshold cryptography servers in the network
φ	threshold
$< M >_{SK_A}$	the message M signed by the private key of node A
$\{M\}_{key}$	the message M encrypted by the key key
$SA_{A,B}$	security association shared by node A and node B

Message

- General -

M	message
M'	modified M
$< field1, field2, ..., fieldN >$	message composed of the fields $field1$, $field2$, ..., $fieldN$
M_Fix	the fixed fields in the message M
M_Var	the variable fields in the message M
$HELLO^p$	the previously received HELLO message
$HELLO^A$	the previously received HELLO message generated by node A
TC^A	the previously received TC message generated by node A

- Message field -

id	message identifier
HC	hop count
SN	sequence number
TTL	Time To Live
$metric$	the known shortest distance to a destination

- Message exchange -

$A \rightarrow \mathcal{N} : M$	node A broadcasts the message M to its direct neighbors
$A \rightarrow B : M$	node A unicasts the message M to another node B
$A \rightarrow \mathcal{N}(B, C, ...) : M$	node A broadcasts the message M to its direct neighbors including nodes B, C, ...,
$A \rightarrow * : M$	node A broadcasts the message M to all nodes in the network

Node

A, B, C, E	nodes
X	misbehaving node
X_1, X_2, ...	misbehaving nodes
D	destination node
I_i	the ith intermediate node on a route
S	source node
$Interface_A$	one of the interface addresses of the node having IP_A as main address
$n_1, n_2, ..., n_N$	the nodes in an N-node network
IP	IP address
IP_A	the IP address of node A
$IP_{A;j}$	the IP address of node A for the cycle c_j
MAC_A	the MAC address of node A
$\mathcal{N}_A$	the direct neighbors of node A

Network and routing

U	upper bound of the number of nodes in the network
N	number of nodes in the network
ca	number of the cooperating internal attackers (also called compromised nodes) in the network
ta	total number of cooperating attackers in the network
d	route length
d_Max	the maximum route length

Trust and supervision

V	the number of observations that are taken into account in the reputation computation
α, β	parameters in the trust model of TRP
$p_{A\to B}(t)$	the experience level that node A has on node B at time t
$p^{+}_{A\to B}(t)$	number of positive experiences of node B that have been observed by node A until time t
$p^{-}_{A\to B}(t)$	number of negative experiences of node B that have been observed by node A until time t
$CD_{A\to B}(t)$	direct trust that node A has on node B at time t
$CDI_{A\to B}$	direct or indirect trust that node A has on node B
$CI_{A\to B}(t)$	indirect trust that node A has on node B at time t
$CR_{S\to S,I_1,...,I_{d-1},D}$	route trust value that the source node S has on the route $S, I_1, ..., I_{d-1}, D$

proof and *certificate*

λ	link state
λ^p	the link state previous to λ
ϕ	no *proof* or *certificate* possible
$from$	advertising node
to	advertised neighbor node
$interval$	the time interval of the creation of a "Link Atomic Information"
$state$	the link state from node $from$ to node to at the time interval $interval$
$< \{"A : state"\},\ T_B(t_i) >_{SK_B}$	*proof* or *certificate* signed by node B showing that at time t_i of node B, B has a $state$ link with node A
$HC_{(A,B,state,i)}$	the hash value which proves the existence of a link of type $state$ from A to B at time interval H_i

Others

$Active - ca - ta$	in the network there are ca cooperating internal attackers (also called compromised nodes), and $ca - ta$ cooperating external attackers
$R_{i;j}$	the root of Hash tree of node n_i in the cycle c_j
UID	unique identifier for each packet (in Network Simulator)
$A \leftrightarrow B$	symmetric link between node A and node B
$A \to B$	asymmetric link from node A to node B

Chapter 1

Introduction

"The Way that can be named, is not the eternal unchanging Way."

– Lao zi (about 500 B.C.)

1.1 Context

With the emergence and integration of mobile computing devices (e.g. cellular phone, notebook, PDA, etc.) and the proliferation of network access points (e.g. airports, railway stations, bars, libraries, etc), traveling people now have more and more possibilities to access networks and information. In the near future, mobile network users will be able to connect to all kinds of networks (GSM, UMTS, Wi-Fi, Internet, etc.), using all kinds of computing devices, anywhere and at anytime. This is also called the fourth generation mobile networking, with which wireless and wired networks will be connected together to form a huge worldwide network, where users will have a maximum mobility thanks to the seamless vertical handovers.
Mobile Ad hoc NETworks (MANETs) [CM99] are specific network configurations that appear in this context. Being self-organized and autonomous, they can either be part of the worldwide network (for example, as part of WiMax [Ohr05] or be connected to Internet via a gateway), or be independent and consist of only mobile nodes that communicate without any infrastructure. Thus, they can be setup on-demand even for remote areas and physically delimited perimeters. They can provide an important means of achieving the ubiquitous network utilization thanks to their ease of deployment and low cost.
There exist a variety of ad hoc scenarios and applications, such as battlefield, conference, emergency service, etc. MANETs have been firstly used in military applications, including emergency rescue activities when the conventional infrastructure-based communication facilities are destroyed due to war, earthquake, or hurricane, etc... They can also be used in residential zones providing an alternative communication means for mobile (and also for fixed) users along highways, university campuses, etc... Note that diverse scenarios can be different in terms of network dimension, node capacity, hostility of environment, Quality of Service (QoS) and security requirements, etc...

The ad hoc routing, which is completely different to that of the traditional networks where dedicated routers are stationary and responsible for forwarding data for end nodes, should be executed on each node in the network in order to ensure a maximum availability of the routing service. Therefore, normally every node in ad hoc networks is also a router. MANET nodes will both generate application traffic and carry out network control and routing protocols.
However, the changing topology and connectivities, network partitions, high error rate, interferences, collisions, and bandwidth and power constraints are issues in the design of (routing) protocols for ad hoc networks.

1.1.1 Secure routing in mobile ad hoc networks

Security is one of the important issues in MANETs especially for the security-sensitive applications such as battlefield. Indeed, pure ad hoc networks are known for their lack of organization, planning and configuration, thus generally being considered difficult to secure.
In traditional networks, it is sufficient to protect and authenticate the dedicated routers in order to secure the routing. Nevertheless, to ensure the *routing security* in MANETs, every node should be responsible for their routing behaviors and also be attentive to the routing behaviors of other nodes, and both the authenticity of topology discovery and the correctness of data forwarding should be ensured.
Our work in this thesis focuses on the routing security, and aim to achieve the above objectives. As a first step, we discuss in the next subsection why the security mechanisms designed for the traditional networks cannot be directly applied to ad hoc networks, and what are the new requirements of the secure routing in MANETs.

1.1.1.1 Why the traditional security mechanisms cannot be directly applied to the MANET routing

We present in the following the critical issues of MANETs that prevent traditional security solutions for wired and cellular networks from being directly applied, and the new requirements of the MANET secure routing:

- We point out that nodes in MANETs are easier to be *compromised* than nodes in traditional wired networks due to their wireless and mobile nature: "wireless" makes communication exposed to the public, and "mobile" makes nodes difficult to be physically protected (for example, easier to be stolen).

  Since compromised nodes cannot be detected with a simple authentication, key schemes alone are not sufficient to resolve the problem, and more sophisticated security mechanisms should be designed. Even though in wired and cellular networks the same problem can also be encountered, it is more challenging to consider security solutions against compromised nodes specific to ad hoc networks. This is because, first, in MANETs some very sophisticated attacks, such as wormhole attacks [HPJ03], can be committed by the colluding compromised nodes; second, in MANETs since the routing depends on every node, the compromised nodes should be detected in every step of routing.

- *Heterogeneous node capacity and battery-based nodes* are also basic issues in MANETs. Some nodes may have constrained lifetime, computational power and memory capacity, and sometimes nodes may be greedy in order to gain more resources.

  Security mechanisms for wired networks are often designed for strong capacity nodes. Since some ad hoc nodes are not able to afford even the asymmetric cryptographic operations, the traditional solutions are not adequate. MANETs require new solutions which are efficient in terms of computational power, energy cost and traffic overhead. In addition, the solutions should also be fair in the consumption of resources.

- For ad hoc networks to work properly, we also need to counteract the *lack of cooperation* between nodes, since MANET nodes have the tendency to be selfish to preserve their constrained resources (bandwidth, battery, etc.). Unfortunately, it is easy to be selfish in MANETs where a node can simply refuse to forward data packets for the other nodes to escape from its routing duty. The routing availability can also be damaged [Mic04] especially when selfish nodes are numerous.

  The problem of selfishness does not exist in traditional networks where nodes do not depend on each other but rely on dedicated routers and servers to ensure the routing functionality. Therefore new mechanisms should be designed for ad hoc networks to ensure the cooperation of all the entities which make use of the MANETs.

- The *lack of organization* can also have considerable impact on the design of security mechanisms for MANETs. For example, since nodes may not know each other when the network is established, a-priori trust relationships may not exist; since there may not be a central key server, key distribution and key management (key revocation) could be difficult to realize; and since nodes may be free to join and leave MANETs at any moment, the network membership could be hard to manage. The above problems generate serious difficulties in guaranteeing the security of ad hoc networks. For example, the spoofing attacks, which are elementary attack operations of many more complex attacks, are difficult to avoid in an open[1] environment.

  Security solutions in traditional networks often rely on either a-priori trust relationships or trusted third part authorities. They use either symmetric or asymmetric cryptographic primitives to authenticate nodes and to secure their data exchanges. However, without infrastructure (central server) nor organization, ad hoc networks need to realize the key and trust management in a more distributed and self-organized way.

[1] Some ad hoc scenarios are in an *open* environment where nodes come from different organizations/places and do not know each other in advance, others are in a *managed* environment where only authorized users can participate in the network.

- Another critical problem in the MANET routing is the *mobility* of nodes that makes neighborhood and topology of ad hoc networks unstable. Thus, it is not easy for a node to be sure of the correctness of the received routing information. Attackers can thus forge and diffuse incorrect topology information for the realization of their attacks. As a consequence, an unsecured ad hoc routing protocol can hardly ensure the correct routing service if there is an attacker in the network. Moreover, attackers can also be mobile to cause more problems in MANETs. For example, they can attack with a spoofed or a bogus identity to avoid being identified, and then move to another place and restart the same attack using another identity, and so on.

  Wired networks have less mobility, and cellular networks usually use infrastructure to manage the mobility of nodes, so their mobile management solutions are not suitable for the mobile management in ad hoc networks. Secure routing discovery protocols should be developed for MANETs in order to exchange reliable neighbor and topology information between nodes.

- The wireless interface, in most cases the *radio interface*, presents numerous problems in the ad hoc routing. For example, due to the radio broadcast nature, the eavesdropping attacks are trivial in the MANET routing; the *hidden node problem*, with which two transmitters that could not hear each other send to the same receiver at the same time, can cause collisions; and the *exposed node problem*, with which nodes in the transmission range of the sender of an on-going session are prevented from making transmissions, can waste the bandwidth of the network. Other problems such as packet loss, transmission errors, unidirectional links, jamming attacks, interferences, signal attenuation, etc... exist also in the ad hoc routing.

  Since traditional security solutions usually require reliable message exchanges, they may not be adequate for ad hoc networks. MANETs require fault tolerate security mechanisms having low overhead.

In this thesis, we treat in particular the problems caused by compromised nodes and the performance issues in ad hoc secure routing mechanisms. We also take into account the problems generated by the selfish nodes and the issues caused by mobility and the lack of organization. We leave the issues caused by the radio interface for future work.

1.2 Motivation

To study the routing security of MANETs, we first need to have a global view of the vulnerabilities against the MANET routing layer. Currently some related works such as [HPJ02, MM04] showed some classifications of attacks in ad hoc networks, and some others studied the attacks against some specific routing protocols. We believe that it is necessary to look for a formal/semi-formal analysis method that is able to analyze the vulnerabilities based on a generic view of the ad hoc routing protocols. Next, we need to determine the scope of vulnerabilities to take into account.

Traditional security mechanisms are not adequate to ensure the security in ad hoc networks. Thus, after the determination of the vulnerabilities that we should treat, we study the new ad hoc secure routing mechanisms and protocols proposed in the literature, for example [HJP02, RACM04, PH02, HBC01, MGLB00, ACL+05]. We study especially the mechanisms that are commonly used in these works such as watchdog [MGLB00], with which nodes are able to detect misbehaviors committed by their neighbors. We believe that watchdog lacks authentication and an efficient storage scheme.
Since some previous studies showed that reactive routing protocols are better suited for ad hoc networks because they generate less control overhead and manage the mobility in a more efficient manner, we quest security solutions for reactive routing protocols. We consider that the cryptographic measures alone are not sufficient to counteract many of the security problems caused by compromised nodes. Therefore, we use some additional mechanisms, such as a reputation system, to identify the compromised nodes and to isolate them from the routing.
We also investigate the security issues in proactive protocols which are especially useful for the scenarios and applications which require a low routing delay. We find that some security mechanisms such as ADVSIG [RACM04] proposed in recent researches for proactive routing protocols are at the price of significant traffic and computational overhead, which may be undesirable for the ad hoc networks with limited bandwidth and processing power. Thus, we intent to lighten some of the mechanisms/protocols without reducing their security level.

1.3 Contributions

The first contribution in this thesis consists of accomplishing a complete threat analysis on the vulnerabilities against the MANET routing layer. To realize this analysis, we use the attack tree model [Sch00], in which threats are classified according to their objectives. An attack tree is composed of a common attack objective (root), some attack sub-objectives (intermediate nodes), and many attack mechanisms (leaves). This analysis presents two advantages. First, to counter the attacks which have a specific attack (sub-)objective, we can just prevent all the attacks listed under the subtree of the (sub-)objective. Secondly, since the tree should contain a complete picture of the MANET routing vulnerabilities, it is sufficient to instantiate the tree under a routing protocol in order to know the vulnerabilities of the protocol.
The second contribution realized in this thesis is the proposition of a secured watchdog mechanism. This mechanism guarantees the authentication in the watchdog supervision by using a broadcast message authentication scheme. In addition, it also provides an efficient memory consumption scheme without loss of the capacity of misbehavior detection.
The third contribution suggests a secure routing protocol named TRP (Trust-based Routing Protocol) that is based on the reactive Dynamic Source Routing (DSR) protocol [JMH04]. TRP uses a trust model to establish trust relationships between nodes, in such a way that misbehaving nodes will be progressively recognized by

benign nodes and then be isolated from routing. Thus, with only a few additional but simple cryptographic operations, TRP achieves the secure routing for both the topology discovery and the data delivery in MANETs. Moreover, TRP can also reduce the control overhead for trust value exchanges, since it integrates them into the routing control messages. Finally, TRP is simulated under attacks with and without the secured watchdog SWAN that is previously proposed.
The last contribution in this thesis concerns the security of the proactive Optimized Link State Routing protocol (OLSR) [CJ03]. It presents two lightweight mechanisms, Hash Proved Link State (HPLS) and Securing TC (TCSec), to prevent forged routing information from being injected into a OLSR network by compromised nodes. Redundant routing information is used in these mechanisms to check the validity of routing entries, thus to ensure the correctness of topology information. We also compare via simulations their performance with ADVSIG (ADVanced SIGnature) [RACM04], which is a main security protocol proposed for OLSR, to illustrate the performance improvements of our secure protocols.

1.4 Thesis organization

This dissertation is organized in seven chapters.
Chapter 1, "Introduction", also the current chapter, provides a global view of the thesis. It introduces the context in which the work in this thesis is realized, the reasons for which the security solutions designed for traditional wired and cellular networks cannot be directly applied to ad hoc routing protocols, and the newly exposed requirements in the design of security mechanisms for ad hoc routing protocols. It also presents the motivations, the contributions and the organization of the thesis.
Chapter 2, "A classification of the threats against the ad hoc network layer", provides firstly a brief introduction of the ad hoc networks, and then studies the routing vulnerabilities in MANETs. It classifies the different threats against the routing layer of MANETs into an attack tree, and then instantiates the tree to find out the vulnerabilities presented in the DSR and OLSR routing protocols.
Chapter 3, "Security mechanisms for the MANET routing", offers an overview of the main security solutions designed for the ad hoc routing. It discusses the design requirements and some security mechanisms for each of the three main research axes in the domain: key and trust management, secure routing and cooperation enforcement. In the end, it discusses our research considerations for the following chapters.
Chapter 4, "SWAN: A Secured Watchdog for Ad hoc Networks", proposes a Secured Watchdog mechanism for Ad hoc Networks (SWAN). It describes the assumptions and the scheme of SWAN as well as the system requirements and the security proprieties of SWAN. It also provides a discussion on the possible optimizations, issues and application range of SWAN.
Chapter 5, "TRP: A Trust-based Routing Protocol for ad hoc networks", proposes the Trust-based Routing Protocol (TRP) that is a secure reactive routing protocol

based on DSR. It presents the reputation system that is used in TRP, and the TRP routing scheme for both the topology discovery and the data delivery routing phases. Finally, it shows the applicability of SWAN for TRP and some simulation results.
Chapter 6, "HPLS and TCSec: Securing OLSR", studies at first the ADVSIG protocol which is a main security solution designed for the OLSR proactive routing protocol. It then suggests an improvement of ADVSIG named ADVSIG$^+$ to counter a security flaw found in ADVSIG. In the following, it proposes two lightweight schemes called respectively Hash Proved Link State (HPLS) and Securing TC (TCSec) to secure the OLSR protocol and to improve the routing performance of ADVSIG$^+$. In the end, it shows also some simulation results.
Chapter 7, "Conclusion", concludes the dissertation with a review of the realized work, some guidelines to design a new ad hoc routing protocol secured from scratch, and some future research directions.

Chapter 2

A classification of the threats against the ad hoc network layer

> "Never harbor the intent to victimize others; but never let guard down against being victimized."
>
> – *Hong Ying Ming « Cai Geng Tan » (about 1570)*

2.1 Introduction

In the early 1970s, mobile ad hoc networks were first introduced and studied by the U.S.A for military usage. Then, their applications have been largely extended. Today, some standards have been defined. For instance, HiperLAN [Net97, Net98a, Net98b], Bluetooth (or IEEE 802.15.1) [bbb05] and IEEE 802.11 [Soc05] (or Wi-Fi for Wireless Fidelity) in mode IBSS (Independent Basic Service Set). Some of them are successfully commercialized.

Due to their nature, ad hoc networks are exposed to a large number of security threats, especially at the routing layer. Studying the threats towards the ad hoc routing layer will permit us to recognize the potential attacks, and then build a security environment to satisfy the security requirements of specific ad hoc applications. Several related work, such as [ZA02, SDL+02], have attempted to classify the vulnerabilities of ad hoc routing protocols. However, these classifications do not address all vulnerabilities of ad hoc networks, and deal only with one or few specific ad hoc routing protocols.

In this chapter, we intend to approach a more complete classification which treats a large number of vulnerabilities based on a generic view of the ad-hoc routing protocols. But our classification only considers threats against the network layer and ignores the ones that target the Physical/MAC layer.

To achieve our goal, we have employed an analysis method named *attack tree* [Sch00]. The method allows us to depict a generic attack tree, which could be an easy and extensible tool to analyze security features of any MANET routing protocol. By instantiating the tree for a specific routing protocol, we can find the flaws of the protocol with respect to an attack objective.

This chapter is organized as follows: ad hoc networks are introduced in section 2.2, and section 2.3 discusses several threat classifications in the literature. Our own classification considerations are presented in section 2.4. In section 2.5, elementary operations of attacks are described. Section 2.6 details our attack tree. Two instances of our attack tree are presented in section 2.7. Finally, the chapter is concluded by section 2.8.

2.2 Background: Ad hoc network

The goal of this section is to provide readers a background about MANET. Concretely, we review the technologies and the existing routing protocols that can be employed by MANET networks.

2.2.1 Physical and media access control layers

802.11

IEEE [oEE] has been working on the specifications for wireless Ethernet technologies since 1996. Actually, it has issued a series of standards in the efforts to enable the communication in a wireless LAN. The current standards include the 802.11 standard and various 802.11 extensions, such as 802.11a, 802.11b, 802.11g, etc...
Besides the infrastructure-based mode where nodes communicate via access points, 802.11 has the IBSS mode, aka the *ad hoc mode*. Within this mode, nodes communicate in a peer-to-peer way.
At the physical layer, 802.11 supports both radio and infrared medium, but infrared medium is much less used. 13 radio channels can be used in 802.11, but only 3 channels can be used simultaneously due to interference. Three radio spread spectrum modulation methods are defined in 802.11. They are respectively Frequency Hopping Spread Spectrum (FHSS), Direct Sequence Spread Spectrum (DSSS) and Orthogonal Frequency Division Multiplexing (OFDM). FHSS is not really used in practice, while DSSS is widely used in 802.11b, and OFDM is used in both 802.11a and 802.11g.
At the MAC layer, 802.11 mostly uses Distributed Coordination Function (DCF)[1] [Soc05] to coordinate communication between different nodes. DCF uses in its turn Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) with optional Request-To-Send - Clear-To-Send (RTS/CTS) to control the media access and the collision avoidance in 802.11.
Because of its simple utilization, 802.11 is widely deployed.

[1]The other mode called Point Coordinator Function (PCF) could only be used in the infrastructure mode.

Bluetooth

Bluetooth [bbb05] is a technology that enables the wireless communication among electronic devices within a short range. It is also specified in IEEE 802.15.1. In Bluetooth, a set of devices sharing a common channel (a common bandwidth) is called a piconet. Within a piconet, the device at the center performs the role of master and all other devices operate as slaves. Up to seven slaves can be active and be served simultaneously by the master. In some usage scenarios, however, devices in different piconets may need to communicate with each other. For this, Bluetooth defines a structure called scatternet to facilitate inter-piconet communication. A scatternet is formed by interconnecting multiple piconets. As many as 10 piconets can overlap to form a scatternet, linking up to 80 Bluetooth appliances in one network. Today Bluetooth is widely used in wireless headsets, wireless keyboards and mouses for computers, etc.

HiperLAN

High performance LAN (HiperLAN) [Net98a, Net98b, Net97] is a family of European Telecommunications Standards Institute (ETSI) standards for WLAN. HiperLAN/2 employs OFDM to modulate data, and Time Division Multiple Access/Time Division Duplex (TDMA/TDD) to control the media access. Even though the HiperLan/2 offers a data rate up to 54Mbps, HiperLAN networks are not commercialized.

Sensor networks

Wireless sensor networks are often deployed to monitor the environment or systems. They are usually composed of a large number of resource-restrained sensors which communicate through an air interface. Actually, sensor networks can be considered as ad-hoc networks since some of them are infrastructureless. However, due to the inherent limitations of sensor nodes, the design of security solutions for sensor networks is different from that for normal ad-hoc networks.

2.2.2 Ad hoc routing layer

In a wireless mobile ad hoc network, two communicating nodes are not always in each other's direct transmission range. Therefore, it is necessary to design multihop routing protocols. An ad-hoc routing protocol should satisfy the following performance requirements:

- Establish routes between nodes in a totally distributed way.
- Rapidly adapt to frequent topology changes, especially dynamically handle broken links at real time.
- Have a low overhead as well as a low consumption of energy.

There are three types of unicast ad hoc routing protocols. They are respectively *proactive*, *reactive* and *hybrid* protocols.

- *Proactive:* Also called "table-driven". Nodes periodically exchange routing control messages to keep their routing information updated. Thus, they are able to perform routing functionalities with no latency, at the expense of bandwidth and energy.

- *Reactive:* Also called "on-demand". Nodes do not exchange routing information until there is a flow to be sent but no route is available. To find a route, sender broadcasts a Route REQuest (RREQ) message, hoping it will reach the destination. Then the destination sends back at least one Route REPly (RREP) message to the sender, using the route which permits the RREQ to reach it. As a result, a route is established between the sender and the receiver. This procedure creates a delay and a burst routing overhead. However, reactive protocols are generally considered to have less overhead than proactive ones. When forwarding data, an intermediate node will send back a Route ERRor (RERR) message to the sender if there is a broken link.

- *Hybrid:* A hybrid approach comes as a compromise between proactive and reactive schemes. The main idea is to allow the routing protocols to initiate the route determination procedure on-demand, but at limited search cost. In such types of protocols, each node maintains the topology information within its zone (coverage area) in a proactive approach, while it discovers the route on-demand for any node outside its zone. The expected advantage from this approach is the scalability improvements. However, this also introduces some additional complexities.

In the rest of the thesis, we will further distinguish two phases in each routing protocol: the *topology discovery phase* in which the topology information is exchanged, and the *data forwarding phase* in which data are delivered from sources to destinations. According to the definition, all routing control messages are sent within the topology discovery phase, including the route maintenance messages such as Route ERRor (RERR).
In the following, two representative MANET routing protocols, Dynamic Source Routing protocol (DSR) [JMH04] and Optimized Link State Routing protocol (OLSR) [CJ03], are introduced. For the other routing protocols, interested readers can refer to [hNC] for more information.

2.2.2.1 The Dynamic Source Routing protocol (DSR)

The reactive DSR protocol [JMH04] operates according to the source routing algorithm. It defines three basic control messages: RREQ, RREP and RERR. In order to send a data packet to a receiver, the sender searches at first in its routing cache

whether there is a complete route to the receiver. If there is an appropriate route, it will be put into the packet's header before the packet is sent out, and any forwarder node has to relay the packet according to it. However, if there is not such a route, the sender broadcasts a RREQ to look for it. Upon receiving the RREQ, the receiver or an intermediate node which has a route towards the receiver, sends back to the sender a RREP containing a/the route. Then, the sender can start to send data. During the data transmission, if an intermediate node finds a link interrupted in its downstream direction, it notifies the sender by sending him a RERR.
In the effort to improve the performance and the dynamism of DSR, a lot of optimizations have been integrated into DSR. For example, by using the promiscuous mode (with which nodes can receive all the messages passing through their neighborhoods), a mechanism named "promiscuous listening" allows nodes to collect topology information from any packet passing by. Another mechanism called "packet salvaging" permits intermediate nodes to modify a route in use when there is a broken link, instead of systematically sending back a RERR. However, due to the lack of security considerations, most of these performance-improving optimizations can be easily exploited by malicious nodes, thus rendering network vulnerable. Therefore they are often deactivated when we secure DSR.

2.2.2.2 The Optimized Link State Routing protocol (OLSR)

OLSR [CJ03] is a proactive routing protocol developed by Institut National de Recherche en Informatique et en Automatique (INRIA). It is already an Internet Engineering Task Force (IETF) Request for Comments (RFC), and its second version is under development. The protocol is an optimization of the classical link state routing algorithm (e.g. the Open Shortest Path First protocol [Moy98]) tailored to the requirements of a mobile wireless LAN. Its key concept is MultiPoint Relays (MPRs), which are selected nodes that forward broadcast messages during the routing flooding process. This concept substantially reduces the message overhead as compared with a classical flooding mechanism where every node rebroadcasts each message when it receives the first copy of the message. Moreover, in OLSR link state messages are generated only by nodes elected as MPRs. Thus, a second optimization is achieved by minimizing the number of control messages flooded in the network. As a third optimization, an MPR node broadcasts only links with its MPR selectors. Hence, contrary to the classic link state algorithm, only partial topology information is distributed into the network. OLSR is particularly suitable for large and dense networks as the technique of MPRs works well in this context.
OLSR has mainly two types of routing control messages: HELLO and Topology Control (TC). HELLO is a local message (with $TTL = 1$) in charge of link sensing and MPR selection. By receiving HELLO messages, a node can be sure of its asymmetric and symmetric one-hop neighbors, symmetric two-hop neighbors, and whether it is chosen by neighbors as MPR. TC is a broadcast message that is used by nodes to declare their MRP selectors. Based on the received HELLO and TC

messages, a node can calculate its routing table and decide a best (shortest) route for each reachable destination.

At the beginning, the research efforts for MANETs were focused on performance aspects. Later, along with the discovery of numerous vulnerabilities in MANETs, the security of ad hoc routing protocols has in its turn become an important research topic. Below we reveal the security vulnerabilities of mobile ad hoc networks and the possible classifications, before presenting MANET security solutions in the following chapters.

2.3 Existing vulnerability classifications

Hu et al. [HPJ02] supposes that there exist some access control mechanisms that allow some nodes to enter the network (these nodes are called *internal nodes*) while refusing the others (these nodes are called *external nodes*). Based on the assumption, the attack possibilities against MANETs are measured by using the term "Active-ca-ta", where ca is the number of cooperating internal attackers (also called compromised nodes), and $ca - ta$ is the number of cooperating external attackers. Otherwise, the term "Passive" is used to indicate that there are only passive attackers, and "Active-VC" means the contrary, i.e., there are so many active attackers that the network is under their control. This measure system is helpful since it clearly tells us the relationship between the number of attackers and the attacks. Thus, it is also used by us in this thesis to show the attack possibilities.

Different to the above work, in [MM03] Michiardi et al. do not emphasize on attackers but on attacks. They distinguish between *active attacks* and *passive attacks*. Active attacks correspond to the attacks that may require a non negligible amount of energy and are carried out by nodes with objective to compromise normal network operations. Passive attacks, on the other hand, are performed by selfish nodes with main objective to save energy. This classification is useful since it distinctly defines the two main objectives of the MANET misbehaving nodes.

Closer to our approach, in [MM04] Murthy et al. present an attack tree model. This attack tree model also distinguishes passive attacks from active ones, and the active attacks are further divided into external and internal attacks. Moreover, all the active attacks are classified according to the network layer [iso94] on which they could happen. The whole classification is quite clear. However, to our opinion, passive attacks can also be both external and internal, and the tree is not yet very complete.

2.4 Discussion

Most classifications presented above do not make a clear distinction between attack objectives and attack mechanisms (i.e. the methods to achieve the objectives). It is

not very surprising, since the nomination of the attacks is not clear from this point of view: some of the attacks are named according to their objectives while others are named according to their mechanisms.
For example, attacks known as *Byzantine attacks* [AHNRR02] are defined as the attacks launched by compromised attackers. They use diverse mechanisms/behaviors to disrupt the system, such as blackhole, wormhole, loop, etc... However, all the Byzantine attacks have a same objective which is to decrease the network performance but still make the network appear to work normally from the viewpoint of benign nodes.
Some other examples such as impersonation, the *sybil attacks* [Dou02] (see section 2.5) or the *wormhole attacks* [HPJ03] (c.f. section 2.5) are defined according to their mechanisms. Their goals could be diverse, for example, sybil attack can be used in data disclosure or in performing "voting attacks" in a reputation system.
We also note that an attack mechanism such as impersonation may serve multiple objectives, and an objective may also be achieved by different mechanisms. We should reflect these relationships in our tree (c.f. section 2.6).
Assumptions on ad hoc networks (existence of trust model, application requirements, choice of routing protocol, key generation and utilization, etc.) can also strongly influence the attack possibilities. For example, the attack "cache poisoning" that can happen under source routing protocols has no chance to take place when another type of routing protocol is in use.
Finally, if we employ in MANETs the security mechanisms that are not very well designed, they themselves can be exploited by attackers. For instance, in the case of Ad hoc On Demand Distance Vector routing protocol (AODV) [PBRD02], a *Sequence Number* (SN) field is added to prevent replay attacks. However, an attacker can spoof a victim's identity and send a message with a higher SN than the victim's current one. Then, other nodes will believe that the SN of the victim has been much increased, and they will reject the victim's packets since its SN is lower. Therefore the victim is excluded from the network unless it can again find an appropriate SN. Some other examples of this kind of attacks are:

- Reputation systems that take second-hand information into consideration are often vulnerable to "blackmail attacks": malicious nodes send false accusations or false reputation exchange messages in order to attribute bad reputations to honest nodes. On the other hand, supervision mechanisms, which are often served as a base of the reputation systems, are also vulnerable to impersonation attacks.

- Heavy cryptographic mechanisms may be exploited to raise Denial of Service (DoS) attacks.

- Location-based routing protocols may suffer from the weakness of the Global Position System (GPS) system, etc...

Taking into account the above-mentioned points, we precise in the next section the elementary attack operations in MANET. Later, in section 2.6, we present our com-

plete attack tree for ad hoc routing protocols, in which we emphasize the objectives of attackers, and we limit our analysis at the routing layer.

2.5 Elementary attack operations

We consider the following elementary operations which can be used to perform attacks:

Message interception

In ad hoc networks, by using the promiscuous mode of 802.11, an attacker can easily intercept (eavesdrop) messages sent in his neighborhood. Even though these attacks are in fact situated at the MAC layer, they can help a lot to realize the routing attacks.
Message interception is usually used in the attacks related to information and data disclosure. Packet encryption is usually employed to prevent this operation.

Message recording and replay

An attacker may be able to record a message that is transited by him, or a message that he is able to eavesdrop. A recorded message can then be replayed or be reused to forge a message.
Message replays can be used in the attacks related to performance degradation and/or topology modification, since they may increase the overhead of the network or introduce obsolete routing information into the network. However, they generally can be easily prevented by integrating time information into messages (if nodes are synchronized) and by guaranteeing the integrity and authentication of messages.

Message dropping

This attack may concern both routing messages and data packets. An attacker can drop a message that is sent across him, or a selfish node can be silent when it should send routing messages. For example, a node may refuse to rebroadcast a RREQ, or to send periodic routing information when a proactive routing protocol is in use.
Message dropping can be used to realize the attacks related to performance degradation and topology modification. However, a supervision system can partially detect this operation.

Message alteration

An attacker may modify part of a message that is transited through him (note that most of the secure ad hoc routing protocols assume that intermediate nodes can alter protocol fields of a routing message). For instance, in AODV a data attraction

attack is possible if an attacker decreases the *hop count* (the number of hops to reach a destination) field in a route discovery message.
Message modification may be used in the attacks related to performance degradation and topology modification. As for countermeasures, it is easy to protect the constant fields, but it is relatively difficult to protect the fields that are to be changed during transmission.

Message forging

An attacker may be able to forge a message. For instance, in DSR, an attacker can send a false route error message to invalidate a link or a route.
Message forging is mainly used in the attacks related to performance degradation and topology modification. It can be partially detected by supervision.

Impersonation (or address spoofing)

Unsecured ad hoc routing protocols do not authenticate source nodes of messages. Thus, a malicious node can launch an attack with a spoofed identity (normally by modifying temporarily its MAC and/or IP address). In an extreme case, a malicious node may forge multiple identities to realize the *Sybil attacks* [Dou02].
Impersonation is usually not used solely but works together with other attacks. An efficient countermeasure against this operation is to authenticate messages by means of cryptography, since a message spoofed cannot be correctly authenticated due to the absence of the appropriate key.

Message exchange through private connection

Colluding attackers may be able to communicate through a private connection, such as a wired tunnel. This mechanism allows attackers to bypass normal wireless connections that are often slower, less reliable, less discrete or longer distanced.
The operation may be used in the attacks related to topology modification and performance degradation. The *wormhole attack* is such an example: an attacker records messages and then tunnels them to a colluding attacker. In other words, cooperating wormhole attackers are able to fool benign nodes with incorrect neighbor (topology) information. We present a few efficient countermeasures against this attack in section 3.4.4.1.

2.6 Attack tree

Attack tree [Sch00] is a formal, methodical way to describe the security of systems. Basically, it represents the attacks against a system in a tree structure, where the root of the tree is an attack goal, a leaf of the tree is an attack mechanism, and an intermediate node of the tree is a so-called subgoal (a more specific goal).

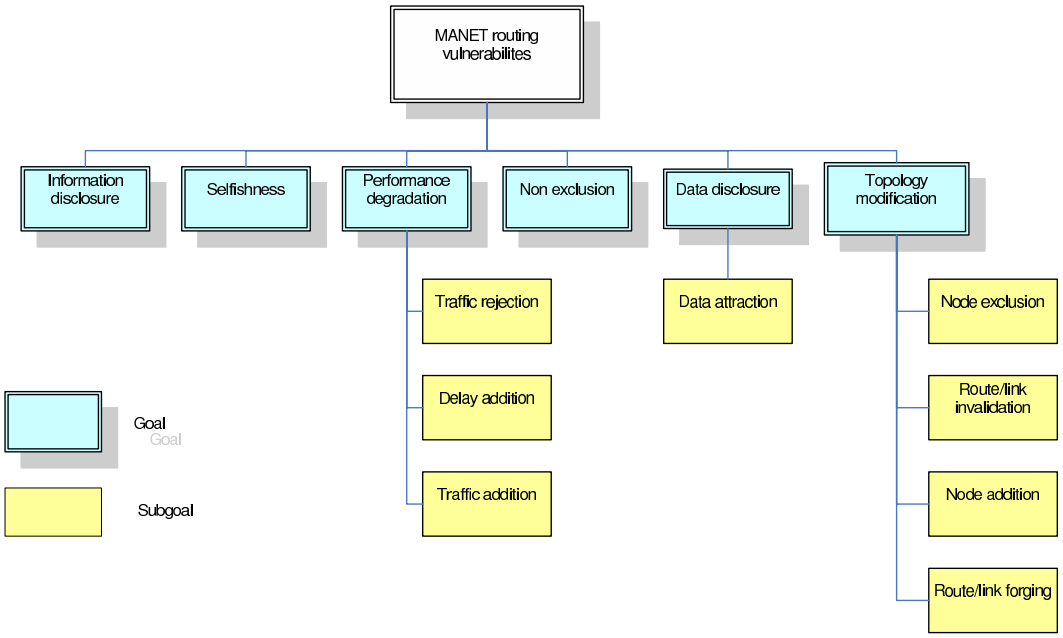


Figure 2.1: Ad hoc network layer threat tree

To establish an attack tree for MANETs, we start with some basic attack goals, and then we refine the basic goals into sub-goals along several paths of the tree. New attack variants can be easily integrated by appending them under the appropriate tree node.

According to [Sch00], the MANET attack goals can be classified into a tree structure as shown in figure 2.1. Due to the lack of place, our tree only presents the different goals and subgoals, while ignoring the methods.

The six main attack goals considered in our attack tree are introduced in the following six subsections, where we also detail their possible subgoals and mechanisms. Our attempt is to apply the attack tree model as a new methodology to the vulnerability analysis of ad hoc networks.

The main advantage of attack tree model is that such a tree can be instantiated for each (MANET routing) protocol, thus providing a good vision of the vulnerabilities of the protocol. Another important advantage of the analysis is the emphasis on the attack goals. We can know at a glance the attacks that are desired by the attackers with a certain attack goal, thus knowing the possible solutions to protect the network from being threatened by such attackers.

In the following, we also distinguish different attacks depending on the types of routing protocols they could be applied to, because not all attacks can be applied to all the routing protocols. There are some attacks which can only be realized with one type (reactive or proactive) of routing protocols, with a certain routing algorithm, etc...

2.6.1 Information disclosure

Information disclosure refers to the attacks that collect information about the network, such as routing information, topology information, node identities, geographic location of nodes, position of important nodes, etc. In some cases, these information are critical for ad hoc networks. For example, in a battlefield MANET, the position of the commanders must not be discovered by the enemies.
Information can be gathered by eavesdropping, if it is not protected. However, the routing information is usually not protected even when a secure ad hoc routing protocol is employed. This comes from the fact that it is difficult to realize information protection in MANETs while still guaranteeing the routing performance. As a consequence, the information disclosure protection is only required by some rare applications, and we do not consider these attacks in this thesis.

2.6.2 Data disclosure

Data disclosure consists in collecting data traffic transited in ad hoc networks. It is obvious that data disclosure can occur with eavesdropping if the confidentiality of data is not ensured.
Otherwise, data disclosure could also be a first step of data tampering attacks if the integrity of data is not well guaranteed.
The main subgoal for data collection is *traffic attraction*.

2.6.2.1 Traffic attraction

Traffic attraction is an attack goal with which attackers try to attract data flows towards them by interfering in the routing discovery phase.

Attacks applicable to all routing protocols

- *Vertex cut attack* - Attackers can control the interconnection points (also called bridges) connecting different part of a network and cut off all other links. As a result, all the communication between the parts will pass by them.

- *Wormhole attack* - With this attack, colluding attackers use a private connection to reduce the length of routes passing through them or the propagation time of routing messages, as described in section 2.5. As a result, a route established on a wormhole seems shorter and faster, therefore it has more chances to be chosen for sending data flows.

- *Increasing Sequence Number (SN) or other increasing message identifier* - In many protocols, an attacker can send messages with high SN. These messages are considered fresher than the normal messages. Therefore legitimate routing messages are rejected, and only routing messages sent by the attacker are accepted. Thus, the data routes will pass through the attacker.

This attack is only applicable to the protocols using these message identifiers. Unfortunately, due to the lack of synchronization, many ad hoc routing protocols use these identifiers, and are thus exploitable.

- *Reducing Hop Count (HC)* - This attack works with all protocols using a Hop Count field in their routing messages, for example, AODV and Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) [PB94]. The route choice can be misled by this attack since a forgery route may always appear shorter.

Attacks specific to reactive protocols

- *Rushing attack* [HPB03] - In most reactive routing protocols, in order to avoid loops and find out the fastest routes, each node will only treat the first copy of a RREQ. To limit collisions, each node should add a random emission delay before rebroadcasting the RREQ.

  By deleting the random delay, a malicious attacker can hurry his RREQ message to next nodes and make other RREQs sent by benign nodes rejected. Therefore the routes containing the attacker have more chances to be chosen to delivery data. This attack is particularly harmful because it can be performed by a weak capcacity attacker.

- *Route cache poisoning* - This attack consists in injecting wrong routing information into routing caches of honest nodes. It works only with source routing protocols since only them have route cache.

  The attack is particularly easy to realize when the DSR protocol is in use, since routing information can be learned through "promiscuous listening" (c.f. section 2.2.2.1).

- *Reducing the number of identities in source route* - Indeed, this attack is specific to source routing protocols. An attacker can erase a number of identities in a source route when sending a RREQ or a RREP. Then the route recorded is seem shorter than the real route, and it has more chances to be chosen.

- *RERR dropping* - An attacker may systematically refuse to generate or forward RERRs in order to make routes passing by him always appear valid. Indeed, such an attack could not work when an end-to-end acknowledgment is required, but this is not the case for most of the MANET reactive routing protocols.

- *Declaring a subnetwork* - In AODV, such an attack can be realized by declaring a subnetwork with a high SN. Attacker can then control all the traffic between the subnetwork and the rest of the network.

Attacks specific to proactive protocols

In the OLSR protocol, to attract data flows, an attacker can try to be selected as a MPR (refer to section 2.2.2.2) with the following mechanisms:

- *Declaring forgery neighbors* - Such an attacker can declare inexistent nodes as neighbors in his HELLO messages. Then, according to the MPR selection rules [CJ03], the attacker will surely be chosen as MPR since only he "has" the "neighbors".

- *Showing a high willingness to be MPR* - With the highest willingness, an attacker is sure to be chosen as MPR. Even a second-level willingness can increase its chance to be MPR.

Attacks specific to protocols using security mechanisms

Same distance fraud - In secure routing protocols (such as SLSP described in section 3.4.3.2) which use hash chain to secure an increasing field (such as HC) or a decreasing field (such as TTL), a malicious node is able to keep the field unchanged. This misbehavior can either make the messages through the attacker reach a larger area or make routes through the attacker seem shorter than their real lengths. Thus, it can attract traffic to the attacker.

2.6.3 Selfishness

Selfish behaviors are not really attacks. They are the behaviors of the nodes which do not cooperate with others to guarantee the good operations of the ad hoc routing. In a distributed mobile network, in order to save bandwidth, computational resources and battery lives, nodes have more intentions to adopt selfish behaviors. These behaviors are often the contrary of data attracting attacks, since selfish nodes do not want to forward data. We call the nodes adopting selfish behaviors selfish nodes. When there are many selfish nodes in a network, its routing service availability could be imperiled (c.f. section 3.5).
Selfish behaviors are generally considered as passive behaviors (without any message sent). However, a few selfish behaviors may also include some active actions.
Below, we distinguish between different selfishness behaviors depending on the type of protocols they can be applied to.

Behaviors applicable to all routing protocols

As an intermediate node, a selfish node can perform a *blackhole* (also called *sinkhole*) or a *greyhole* attack during the data forwarding phase. These attacks drop data packets instead of forwarding them.

Behaviors specific to reactive protocols

A selfish behavior can also be:

- *Non participation into the topology discovery phase* - A selfish node drops RREQ and/or RREP messages that it should resend.

- *Modifying routing discovery message* - A node modifies RREQ or RREP messages (with longer route, lower sequence number, etc.) to make the routes passing by them discarded by the source nodes. However, in most cases, this attack can also provide non-optimal routes to attackers, so we do not consider this attack in our thesis.

- *Sending forgery route maintenance message* - A node sends route error messages (even if there is no broken link) in order to avoid forwarding data packets.

Behaviors specific to proactive protocols

In OLSR, a selfish behavior could be *showing its willingness of not being MPR*, since such a node is sure not to be chosen as MPR.

Behaviors specific to protocols using security mechanisms

With a reputation system, when the punishment strategies decide that a selfish node will not be excluded from network but only from routing (in order to prevent the rejection of data packets), a possible selfishness behavior could be *giving a bad reputation to itself*. This is because the selfish node is not excluded from network, therefore it can continue to benefit the routing service as sender or receiver; furthermore, it is excluded from routing, so it will not be chosen as intermediate node and can thus be naturally selfish.

Remark: We could see from this case that the punishment strategies of a reputation system (more generally all the security mechanisms) should be carefully designed according to the security objectives that we want to achieve.

2.6.4 Performance degradation

Performance degradation aims at perturbing the ad hoc routing or causing DoS attacks in MANETs. The attacks in this category can be classified according to the three following subgoals.

2.6.4.1 Data rejection

Data rejection can be used to degrade the routing performance, since it can result in data loss. The following methods may cause data rejection in MANETs:

- *Blackhole (sinkhole) attack* - The attacker drops all data packets passing through it.

- *Greyhole attack* - This is a partial blackhole attack where an attacker partially rejects data packets.

2.6.4.2 Traffic addition

Adding redundant traffic into MANETs can increase the routing load thus decreasing the routing performance. A method to realize a denial of service attack is to overcharge a victim by flooding him a huge amount of traffic/requests.
Moreover, in MANETs, in order to save their battery lifes, attackers may prefer not to consume energy for attacking. Thus, a smart attacker should be able to make other nodes generate additional traffic.

Attacks applicable to all routing protocols

Common mechanisms used to overload MANETs with data traffic are:

- *Data message replay* (c.f. section 2.5).
- *Message loop* - this attack makes messages loop infinitely within network in order to consume network resources. And a smart attacker will avoid himself being implicated in the loops. Fortunately most routing protocols are protected from this attack.

Attacks specific to protocols using security mechanisms

When a secure routing protocol uses the authentication and integrity verifications mechanisms which consume a significant amount of computational resources, attackers can bring down networks by sending a great number of bogus messages.

2.6.4.3 Delay addition

In the following, we list different methods that can be used to add delay to data delivery. They work essentially with reactive protocols:

- *Providing non-optimal route* - By modifying route discovery messages, an attacker can make other nodes use non-optimal routes to delivery data. This attack is also a type of byzantine attacks (c.f. section 2.4).
- *Modifying data packet header* - The arriving time of a data packet could be postponed when it is deliberately modified by an attacker. For example, the attacker can redirect it to another neighbor which may have a worse route to the destination. This method works also with proactive protocols.

2.6.5 Topology modification

In this section, we consider the attacks whose goal is to modify the connectivity of network. We distinguish the four following subgoals.

- *Node exclusion/isolation*: Attackers try to exclude or isolate some benign nodes.
- *Node addition*: Attackers try to introduce forge node identities into MANETs.
- *Route/link invalidation*: Attackers try to invalidate legitimate routes or links.

- *Route/link forging*: Attackers try to inject forged routing information into MANETs.

2.6.5.1 Node exclusion/isolation

When a node is excluded from network, the topology of the network is changed. To exclude a node, attackers can cut off its communication with the other nodes. The following methods can be used to achieve this goal:

Attacks applicable to all routing protocols

Sleep deprivation attack - An attacker uses up the battery of a node by sending it a large number of packets. The node is excluded once it has no energy left.

Attacks applicable to proactive protocols

Sybil attack - Together with spoofing, this attack can be used to disable all links of a node. That is, an attacker can pretend to be all the neighbors of a victim and then invalidates each of the victim's links.

Attacks applicable to reactive protocols

Using a great SN plus impersonation - In AODV [PBRD02], by using a high SN, an attacker can isolate a node impersonated until the victim could find again an appropriate SN.

Attacks specific to protocols using security mechanisms

Blackmail attack - The attack may be applied to a secure routing protocol based on a reputation system that accepts recommendations. Attackers can send wrong accusations against honest nodes, or cooperate to accuse firmly a victim node.

2.6.5.2 Node addition

To modify the topology, an attacker can also "add" nodes to a network by pretending to be many nodes, which is also called a "sybil attack". Therefore non-existent nodes may be introduced into routing tables or route caches if there is not a central server that maintains network members and/or controls network access. Note that Statistically Unique and Cryptographically Verifiable (SUCV) addresses [MC02] (c.f. section 3.3.2.4) are vulnerable to this attack.

2.6.5.3 Route/link invalidation

Links may be invalidated by attackers. And once a link is invalidated, all the routes using this link are also invalidated, and the topology is modified.

Attacks specific to reactive protocols

Impersonate RERR message - An attacker can send out forgery RERR messages by spoofing the identities of nodes on routes. All nodes receiving the message would invalidate the link and change their network topology vision.

Attacks specific to proactive protocols

If a routing protocol only employs symmetric links, then it is sufficient to degrade a symmetric link to an asymmetric one to invalidate a link. This is the case of OLSR, where a link can be invalidated by an impersonated message which declares the link lost.

2.6.5.4 Route/link forging

By forging a link/route, the topology of a MANET can be modified. A route forging attack can be achieved by the following methods:

- With all protocols, one wormhole is also a forged link (c.f subsection 2.5).
- With reactive protocols, a link/route can be forged when impersonation is possible, or when routing messages are alterable. For example, an attacker can pretend to be a far away node and rebroadcast a RREQ, or delete some identities from the header of a RREP of DSR.
- With proactive protocols, by inserting forgery links into routing control messages, receivers of the messages will believe in the existence of the links.

2.6.6 Non exclusion

In fact, non exclusion is not really a malicious attack goal. It is a vulnerability coming from the malicious nodes which do not want to be excluded from network due to security mechanisms. For example, when a reputation system is employed to exclude misbehaving nodes, or when a certain security mechanism is used to cut off links towards attackers, non exclusion behaviors can exist.
An attacker can try to bypass a reputation system by the following methods:

- *Using impersonation* - An attacker impersonates another identity before attacking.
- *Adopting a good proportion of benign behaviors* - Due to the indeterminacy of MANETs, many reputation systems are required to tolerate a percentage (represented by a threshold) of bad behaviors. Thus, to accumulate good behaviors, attackers may exploit this toleration by creating useless but correct traffic among cooperating attackers. Nevertheless, this mechanism consumes energy of attackers.

- *Cooperating to make attacks undetectable* - When the supervision is done only on routes (c.f. section 3.5.3), cooperating neighbor attackers and wormhole attackers are difficult to be detected.

- *Covering up for other attackers* - In a reputation system, a liar is a node which exchanges wrong reputation information with other nodes. A liar can cooperate with attackers to prevent them from being excluded:

  - In an alarm-based reputation system, a liar may refuse to send alarms if he will not be punished for its silence.
  - In systems based on periodic reputation exchanges, a liar may show normal or good reputations for malicious nodes. In an extreme situation where liars are numerous, honest nodes could even be accused of sending blackmails due to their minority.

2.7 Instances of the attack tree

In this section, we establish two instances of our attack tree for DSR and OLSR, respectively in figure 2.2 and figure 2.3. We use these trees to show the vulnerabilities of the unsecured DSR and OLSR.
In the figures, we use different colors to show the goals, subgoals, attacks that we will take into account in this thesis, and the attacks that we do not consider in this thesis. We make choice according to the difficulty and conditions that are required to counter these attacks. For example, we do not consider attacks only realizable with "Active-VC" (c.f. section 2.3).

2.8 Conclusion

In this chapter, we introduced the standards of mobile ad hoc networks, which provide us the basic knowledge to understand this thesis.
Then we presented a classification of the threats that menace the network layer of MANETs. We consider our attack tree as a first step towards a useful frame of a semi-formal security analysis for ad hoc routing protocols. Our main improvement in this work is that we distinguish objectives and mechanisms of attacks. The distinction can help security defenders to easily notice which attacks should be prevented under which security objectives, and which attacks may not be taken into consideration.
Even though nowadays a large number of threats against MANETs are already known, new attacks could still be progressively revealed by researchers or attackers. Therefore it is obvious that our attack tree still needs to be refined, and additional attack objectives and mechanisms should still be added to the tree to make it more complete.
Through our analysis, we noticed that new vulnerabilities may arise due to the introduction of security mechanisms. As a result, we draw the conclusion that

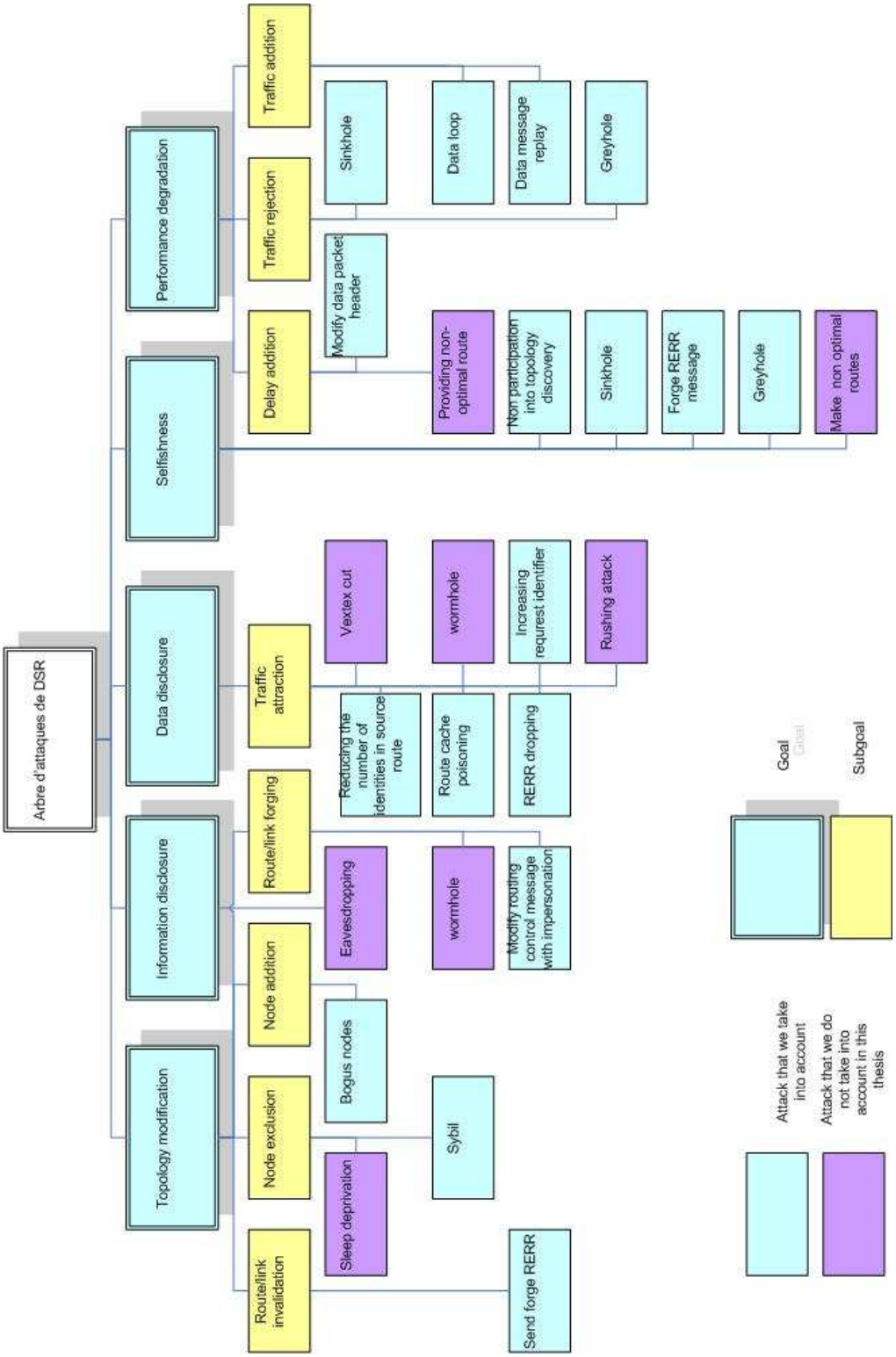


Figure 2.2: Threat tree of DSR

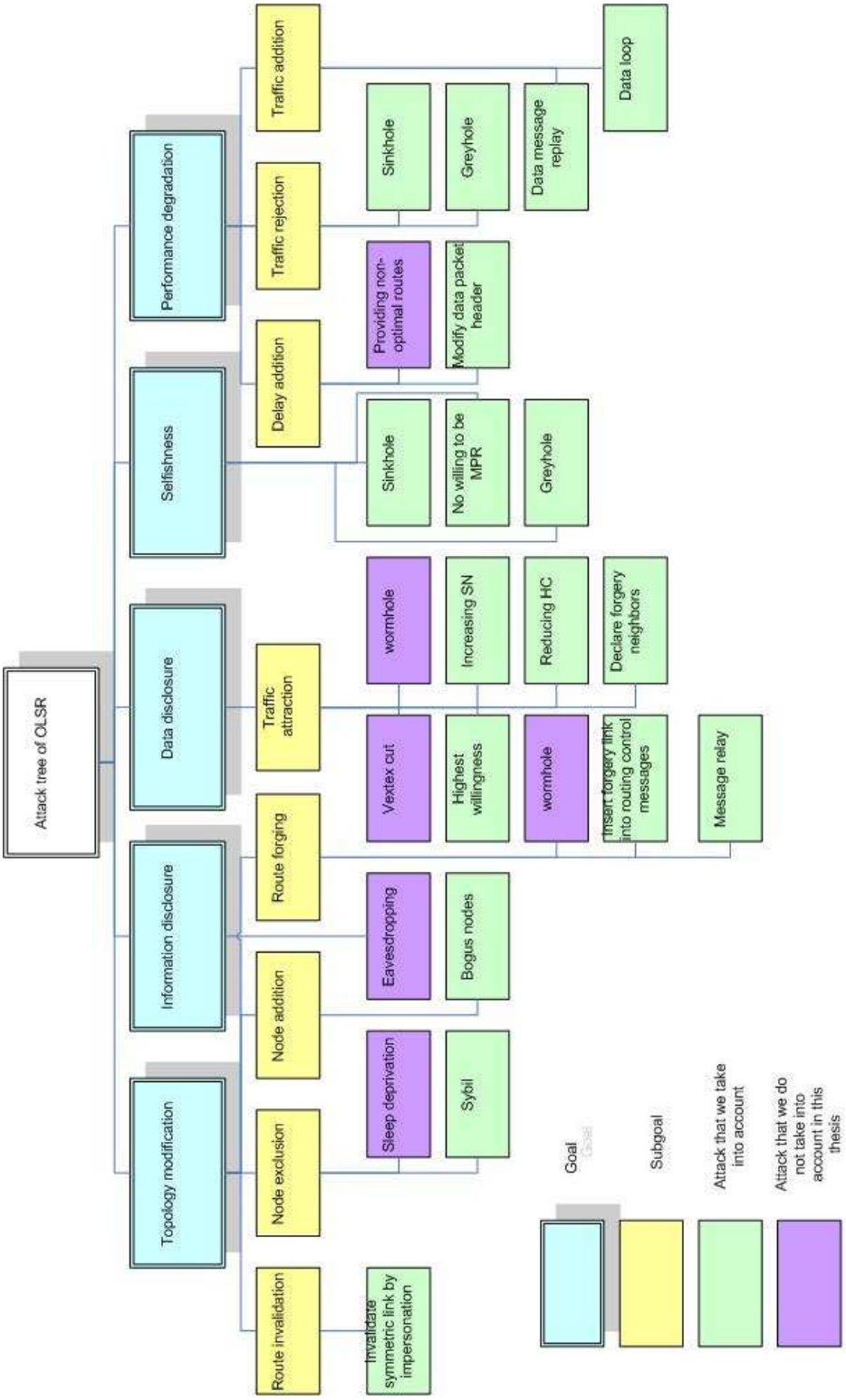


Figure 2.3: Threat tree of OLSR

attentions should be paid to the design of security mechanisms. In addition, the compromise between performance and security should be carefully studied, too.
In this chapter, we also established two instances of our attack tree, for DSR and OLSR respectively. These trees will later be used to analyze our security solutions proposed for these two protocols.

Chapter 3

Security mechanisms for the MANET routing

> "The way ahead is long; I see no ending, yet high and low I'll search with my will unbending."
>
> *– Qu Yuan « Li Sao » (about 340 B.C. - 277 B.C.)*

3.1 Introduction

To address the vulnerabilities discussed in the previous chapter, many secure ad hoc routing mechanisms have been proposed in the literature. Most of them can be classified into the three following categories:

- **Key management** mechanisms deal with identification and all issues regarding keys (establishment, distribution, revocation, renewal and exchange).
- **Secure routing** mechanisms ensure the authentication, confidentiality, integrity and eventually the non-repudiation in the two routing phases: *topology discovery* and *data forwarding*.
- **Cooperation reinforcement** mechanisms fight against selfish behaviors and encourage the cooperation between MANET nodes.

In this chapter, we discuss the representative solutions in the three categories. In addition, we note also that, for any of the MANET security mechanisms, trust relationship is important to be managed. It is the base of the relationships between nodes in MANETs. For example, trust can decide the routing choices in ad hoc networks, if we allow routes to be established only between nodes trusting each other. Reversely, trust relationships can also be influenced by the routing behaviors of the nodes, because a misbehaving node might lose its trust relationships with others. Finally, the evolution regarding trust must also be reflected by the key management: untrusted nodes should not be able to renew its keys.
In most traditional networks, a trust can either be a third-party trust or a direct trust. With a third-party trust, two individuals trust each other via a common

third-party. With a direct trust, on the contrary, trust relationships are established directly between entities themselves. Unfortunately, in a MANET, there may not be a common third-party entity, and nodes may not have a-priori direct trust among them. Even existing trust relationships can be ephemeral due to the mobility and the compromised nodes. This makes the establishment/reestablishment of trust relationships between nodes a key concern in MANETs.
In the rest of the chapter, we present the security mechanisms designed for the three categories mentioned above, and we present a conclusion and show our research considerations at the end of the chapter.

3.2 Notations

We introduce in the following the notations that are used in this chapter in their appearing order.

Notation	Meaning
ns	number of the threshold cryptography servers in the network
N	number of nodes in the network
φ	a threshold
A, B, C, E	nodes
l	the length (in bits) of hash values
$n_1, n_2, ..., n_N$	the nodes in an N-node network
IP	an IP address
PK	a public key
$h(a)$	the hash value of a
$K_{A,B}$	a symmetric key shared by node A and node B
S	a source node
D	a destination node
I_i	the ith intermediate node on a route
s_i	a random seed chosen by node n_i
$\mathfrak{L}$	the length of a Hash chain
$h^j(a)$	a value a hashed j times without key
d	a route length
M	a message
$A \rightarrow * : M$	a node A broadcasts the message M to all nodes in the network
IP_A	the IP address of node A
τ	a time interval
$i\|j$	the catenation of i and j
$h_{key}(a)$	HMAC computed on a value a using the key key
$K_{I_{i_{\tau_i}}}$	the TESLA key of node I_i at time interval τ_i
$A \rightarrow B : M$	a node A unicasts the message M to another node B
SN	a sequence number
id	a message identifier
CA	a CA server
PK_A	the public key of node A

t	a time
e	an expiration time
$cert_A$	the certificate of node A
SK_A	the private key of node A
$< M >_{SK_A}$	the message M signed by the private key of node A
$\{M\}_{key}$	the message M encrypted by the key key
HC	a hop count
$SA_{A,B}$	security association shared by node A and node B
d_Max	the maximum route length
TTL	a Time To Live
τ_i	the ith time interval
t_i	the ith timestamp
TIK_i	the TIK key at time t_i
UID	a unique identifier for each packet (in Network Simulator)
MAC_A	the MAC address of node A
X	a misbehaving node

3.3 Key and trust management

An ad hoc key scheme can be either asymmetric or symmetric. In the first case, it mainly manages a Public Key Infrastructure (PKI), wherein each node has at least a pair of private/public keys. In the second case, it mainly manages symmetric keys, either one symmetric key shared by all nodes of a network, or multiple pairs of symmetric keys each shared by two or more nodes.
The choice between using symmetric and asymmetric key in MANETs often depends on the network scenario (network type, application, etc.). Symmetric keys are better adapted for sensor networks (because they cannot support asymmetric cryptography), or stable and small networks (because it is difficult to manage large numbers of symmetric keys). Asymmetric keys are adapted for networks with a large number of node, or highly dynamic networks.
The cryptographic primitives permit to establish *Security Associations* (SA) [DMMJ98] between nodes. A security association is a relationship between two or more entities that describes how the entities will use security services to communicate securely. This relationship is represented by a set of information that can be considered a contract between the entities. That information must be agreed upon and shared between all the entities that trust each other and participate into a same SA.

3.3.1 Design requirement

When designing a key management scheme for MANET, we should consider the following requirements:

Identified Secrets should be established only with identified nodes.

Distributed The scheme should not depend on any central server, otherwise the server(s) can be the target(s) of numerous attacks. Furthermore, to be totally distributed, it is recommended that all nodes in the network participate in the network key generation process. However, for large networks, this could be too complicate to realize. So scalable approaches based on hierarchical/cluster are suggested in most cases, such as in [BHBR01] and [SA99].

Lightweight It should be lightweight, both in term of protocol exchange and cryptographic operation. This is especially important for the networks with nodes having limited capacity or with proactive protocols.

Key refreshed Generated keys should be relatively strong so that they are not easy to be compromised during the network existence, otherwise they should be refreshed periodically.

Misbehaving node excluded Misbehaving nodes must not be trusted and their keys should be invalidated.

Robust A key management scheme could be itself a target of attacks, so it must be functional even under attacks.

Flexible A leaving node often departs from the network with some network secrets, and a joining node should be rapidly informed of the secrets of the network and establish keys (however the capability of a new joining node depends on the security mechanism employed) in order to communicate with others. Therefore in a discretional ad hoc network, joining and leaving of nodes should be taken into account in the key management.

In the rest of this section, we make a survey of the main solutions suggested for key and security association management in ad hoc networks.

3.3.2 Asymmetric key management

The deployment of traditional PKI system in ad hoc networks is problematic, since such a system needs a Certificate Authority (CA) which is a center server. In addition, to ensure the delivery and revocation of certificates, the CA is required to be permanently online and always accessible by any node. These issues make the traditional PKI inadapted to MANETs.
To get rid of the dependence on CA but keep the advantages of PKI, four kinds of solutions have been proposed. One distributes the functionalities of CA to a number of ad hoc nodes. One uses a trust architecture similar to PGP [S.G95]. And the two other solutions replace certificate by a one-way (hash) relationship between identity and public key for each node. They are respectively introduced in the four following subsections.

3.3.2.1 Distributed certification authority

Instead of using a central CA, we can use the threshold cryptography to distribute the CA functionalities to a number of nodes in a MANET. For example, the private key or the signature function of CA can be distributed to ns nodes in an N-node network ($N \geq ns$), then each of these ns nodes can play the role of a partial certificate authority. We call the information given to each of the ns nodes a *part*. Thus, a number of nodes which own a part will be able to emulate a CA server.
To apply a CA operation, an applicant needs to solicit the authority service from at least $\varphi + 1$ (φ is the *threshold*) nodes among the ns partial servers ($ns \geq \varphi + 1$) [Sha79, SH02]. Each partial server solicited will provide a partial result, and then the results from no less than $\varphi + 1$ servers can be combined with verifications and fault tolerances into a signature which is equivalent to a signature signed by a central server. If ns is larger than $2\varphi + 1$, we see that the network can tolerate up to φ compromised partial CA servers. Note that the *parts* should be periodically refreshed to prevent attackers from compromising more than φ servers.

The Cornell On-line Certification Authority (COCA) [ZH99, ZSvR02], proposition of Zhou et al., requires that $ns \geq 3\varphi + 1$ and $N > ns$ in order to limit the number of partial servers. The private key of the CA is distributed, and it will be reconstructed for each signature at one of the servers. In COCA, applicants can either request a new certificate or update an old one. Moreover, an existing certificate should be revoked if more than φ servers judge that the node which owns the certificate is compromised.
Since ns is limited, COCA may easily protect the partial servers from being compromised. Nevertheless, the protocol has significant overhead and introduces the following disadvantages:

- The exchanges between servers are quite interactive[1]. This is especially true for the *proactive secret sharing* method that is used for the refreshment of the parts. Moreover, to ensure the delivery of the messages, all messages in COCA should be acknowledged.
- Every server has its own pair of public/private keys with which all their exchanges should be signed, including the acknowledgments.
- Multihop routing is used for the key management, creating a loop *routing - key management*.
- For each operation, the private key of the CA should be reconstructed at one of the servers. Thus attackers have the possibility to discover the key if the server chosen is compromised.

Luo et al. introduced another approach [KZL+01, LL00] which defines $ns = N$. During the initiation phase at least $\varphi + 1$ nodes will be given a *part*, and then the other nodes could be initialized on-the-fly by the nodes already initialized. Unlike

[1] Here "interactive" means that there are many message exchanges between non-neighbor nodes

COCA, this approach does not distribute the private key of the CA but only the signature function.
It is assumed that the network is dense enough for all nodes to have at least φ legitimate neighbors which can answer its CA request (the applicant can provide one part itself). This localization of the CA service breaks the loop between the routing and the key management, and makes the protocol non-interactive, unless the applicant does not have enough legitimate neighbor nodes. In this case, two-hop or still faraway neighbors can be solicited. Simulations showed that highly mobile networks can better support the protocol since with mobility nodes have more chances to contact others.
The renewal of certificate should be done with at least φ nodes which trust the applicant, and nodes without legitimate certificate will be isolated because no node will relay their packets. For revocation, a local Certificate Revocation List (CRL) [NDBJ01] exists at each node. Each node is supervised by its neighbors, and its certificate can be revoked if it is found behaving badly. A revoking node updates its CRL and broadcasts the revocation information. Any node hearing no fewer than $\varphi + 1$ accusations against a same certificate should add the certificate to its own CRL to revoke it locally.
The approach of Luo is both distributed and non-interactive. However, if there is not an efficient access control mechanism, attackers can enter the network, make a clan of more than φ members and then revoke and sign certificates. Moreover, in comparison to the solution of Zhou [ZH99, ZSvR02], in this approach all nodes need to be protected from being compromised. Thus the parameter φ should be chosen carefully – if it is too small, the security level will be low; if it is too large, it is difficult to ensure that there are always sufficient legitimate neighbor nodes.

Proposition	Distributed	Requirements	Interactive	Refreshment of parts
Zhou et al.	Partially	$ns \geq 3\varphi + 1$, $N > ns$	Yes	Yes
Luo et al.	Completely	$N = ns$	No	Yes

Table 3.1: Distributed CA solutions

3.3.2.2 Self-organized PKI

Capkun et al. have adopted the idea of *Pretty Good Privacy* (PGP) [S.G95] to MANETs to create a fully distributed PKI without central server [HBC01].
The main characteristic of PGP is the transitivity and the self-organization of trust, which removes the dependency of PKI on a CA server. However, central certificate repositories are still required by PGP. Then, to further remove this requirement, Capkun lets each node own a local repository of certificates.
The *trust transitivity* indicates that if A trusts B and B trusts C, then A trusts C. Following this guideline, each node stores at first the certificates issued by itself as well as the ones destined to it. In addition, it further selects some other certificates and stores them in the residual space of its repository.

Moreover, the mechanism can be more efficient thanks to the employment of the *Heuristic Shortcut Hunter Algorithm*, since a *shortcut* can be used to shorten certificate chains thus reducing both the cryptographic overhead and the possibility of having wrong trusts due to trust transitivity. An alternative way to counteract the forged trust is to combine disjoint multipath certificate chains towards a same node. The *small world* theory [Uni06] argues that any two people in the world can be connected via no more than six degrees of separation. Thus, since the approach is supported by this theory, when two nodes want to communicate without an a-priori trust relationship, they theoretically can always converge their repositories to find the shortest certificate chain between them.
The approach is more appropriate to small ad hoc networks since the convergence (based on the asymmetric cryptography) of long certificate chains could be heavy. Besides, once a certificate is revoked, all chains containing it become invalid and new computations should be effected. Also, it is important to note that we need some initial trusts between nodes for the establishment of certificate chains. Moreover, the trust transitivity is unsuitable especially for the scenarios in a hostile environment due to compromised nodes.

Until now, all the propositions presented in this section employ certificates that provide essentially a binding between node IDentifiers (ID) (often IP addresses of nodes) and public keys. In the next two sections, we will show two alternative solutions which bind naturally an ID with a public key without help of any certificate.

3.3.2.3 ID-based cryptography

With *ID-based cryptography* [BF01], a node decides at first its ID, then hashes the ID to obtain its public key. Afterwards, the node sends a request to a Private Key Generator (PKG) server to request its private key. An online PKG could also be distributed using a threshold cryptography scheme, as suggested in [KKA03], where identities of nodes are even directly used as public keys.
Since the hashing computation cost is significantly lower than that of certificate, the solution is considered as a good candidate for the key management in ad hoc networks. However, the ID-based cryptography scheme still needs further investigation to become mature and be deployed in reality. And no matter whether the PKG server is on-line or offline, distributed or centralized, the control of identities should be ensured.

3.3.2.4 Cryptography-based address

The Statistically Unique Cryptographically Verifiable (SUCV) [MC02] approach is different to that of ID-based cryptography in the way that, instead of generating a pair of public/private key based on a chosen ID, a node generates a pair of public/private keys, and then computes its ID (address) by hashing of the public key. Thus, neither server nor certificate is needed by the approach.
To be able to resist spoofing attacks and to be statistically unique, the hash output (ID) length cannot be too short. In fact, with a perfect l-bit hash algorithm, an

attacker needs on average 2^{l-1} hash operations to discover the corresponding public key, and on average $2^{l/2}$ nodes could generate an address collision [KBC97]. Consequently, IPv4 addresses ($l = 32$) are not sufficiently long, only IPv6 addresses ($l = 128$) are usable by the scheme.
In an unsecured MANET, a misbehaving node may use multiple identities. We call all the identities except the legitimate identity *bogus identities*. Without server to control the identities (addresses) of nodes, bogus addresses can exist along with SUCV which will cause security problems especially for the identification of nodes.
To summarize, we compare the discussed propositions in table 3.2.

Proposition	Certificate	Need server	Characteristic	Revocation
Threshold cryptography	Yes	Yes	Distributed CA	Yes
Self-organized PKI	Yes	No	Transitive trust	Yes
ID-based cryptography	No	Yes	PK = hash(ID)	Possible
Cryptography-based address	No	No	ID = hash(PK)	No

Table 3.2: Asymmetric key management propositions

3.3.3 Symmetric key management

In this section, we describe at first two propositions that aim at establishing a global symmetric key in a MANET. Then we present some approaches to create (or distribute) pairwise keys.
If a network uses a symmetric key shared by all the nodes, it should be attentive to the changes in the network membership, since the modification of the composition of the network will necessitate the regeneration of the key. Otherwise, if the network uses pairwise keys, the same problem does not exist but in a N-node network, nodes are required to store $\frac{N(N-1)}{2}$ keys.
Although much more complicated in terms of management, a pairwise key scheme is considered safer than a global key scheme. This is because, in the case of pairwise keys, compromising one node could not influence a lot the whole network security, while with one global symmetric key, one node compromised signifies that the whole network is compromised. As a result, pairwise symmetric keys are more suitable for scenarios which need a high security level but nodes are relatively weak to support asymmetric cryptography, and a global symmetric key is suitable for MANETs in a safe and stable environment.

3.3.3.1 Password-based authenticated key agreement

Ginzboorg et al. suggested a key agreement scheme in [AG00] for the following scenario: all participants of a conference are in a same meeting room and they trust

each other; they share a password (which is too weak to be a communication key but can be used for the authentication of participants of the network) by some offline means (for example the password could be written on a blackboard in the meeting room); they try to create a strong and global symmetric key to secure all their internal communications.
It is required that with authentication based on the password, all nodes contribute to the generation of the key. Therefore eavesdropping attackers are excluded from the network due to their lack of the password.
Three variants of Diffie-Hellman (DH) key agreement protocol [Res99] are proposed in [AG00], one is a two-part protocol and the others are multipart protocols. In the two-part protocol, nodes exchange their DH public components encrypted by the password, then use challenges to authenticate the communication key. The first multipart protocol is made of $N+2$ steps if the network is composed of N nodes. The first $N-1$ steps consist of computing $\pi = g^{S_1 S_2 \ldots S_{N-1}}$ (g^{S_i} is the DH public part of the node n_i), then the node $N-1$ broadcasts π to all others. Each node authenticates itself to the node N by sending $c_i = \pi^{S'_i/S_i}$ (S'_i is a random number) to it, and the node n_N should subsequently return back $(c_i)^{S_N}$ to each node n_i. After these steps, all nodes should be able to compute $g^{S_1 S_2 \ldots S_N}$ which will be the final key. Finally, the authentication of the key is performed. The second multipart protocol is called *hypercube protocol* since it considers a dm-dimension cube within a 2^{dm}-node network. Thanks to the parallelism, only $dm+2$ steps are needed to establish an authenticated key.
However, this scheme is not flexible with regard to joining and leaving of nodes. Once a node leaves or joins the network, the key has to be rebuilt with a new password, due to the weakness of the password. Furthermore, the work in [Hie01] showed that both the mobility and the topology can influence the performance of the scheme. At last, the approach can be adapted to few other scenarios.

3.3.3.2 Password-based hierarchical key transport

A *smart dust* network is defined as a network composed of a large number of small nodes (for example, sensor nodes). In [BHBR01], a key scheme is proposed for smart dust networks which is somewhat similar to the previous solution since it is also based on a pre-established password and it also generates global communication keys (one key per interval). However, due to the weakness and the important number of the nodes, the DH key agreement protocol is not suitable for the scenario, and a key transport protocol is used instead.
A loose synchronization (which guarantees an upper bound on the maximum synchronization error) is assumed, and it is also supposed that each node has a tamper-resistant device which is able to protect the password and the temporary keys from being compromised.
In such a network, at first, we establish clusters and elect Cluster Headers (CH) which are nodes in charge of communication between clusters during the key establishment phase. Then, it is up to CHs to elect among them a network header who will decide and distribute a communication key to all nodes of the network.

Due to the additional tamper-resistant module, the price and the energy consumption increase for each node. On the other hand, it is found that the so-called tamper-resistant hardware may not be always safe [AK96]. In addition, attackers can even easily be selected as headers if they answer to the CH selection conditions. Thus, this mechanism requires an access control mechanism and a secure password distribution. Finally, this scheme does not provide any way to punish neither attackers nor selfish-nodes.

3.3.3.3 Random key predistribution

Random key predistribution approaches [CPS03, Cha04] involve less overhead on communication and computation. They are usually used with resource-constrained (sensor) networks. The basic key predistribution scheme was introduced by Eschenauer and Gligor in [EG02]. Before entering a network, each sensor receives from a large *key pool* a set of symmetric keys, in such a way that any two nodes in the network can find at least one common key within their key sets.
In [CPS03], an improved scheme is introduced by Chan et al.. It ensures q ($q \geq 1$) common keys between any two nodes, thus communicating peers can randomly choose one of them or use multiple keys at once. Consequently, it will be more difficult for attackers to compromise the communications since it is not easy for them to know the right keys that are used in communications.
In [Cha04], Du et al. use the *node deployment knowledge* to refine the basic key predistribution scheme. Their new scheme takes into account stable neighborhood between nodes, thus being more efficient and using less keys. However, the solution requires that long distance peer-to-peer communications are rare, and nodes are not very mobile.

3.3.3.4 Resurrecting duckling

Resurrecting duckling [SA99] is a key management approach designed for Bluetooth networks [bbb05] (c.f. section 2.2.1). With this scheme, a Bluetooth network is structured into a tree where a parent schedules all transmission of his children. The root of the tree will be the owner of the network, generally a person who owns and controls all his devices.
In short, resurrecting duckling is a hierarchical key transport solution where the keys are given from children to parents. Every child possesses a hardware module with which it can authenticate itself and send a symmetric key to its parent via a physical contact. A parent has nevertheless the right to decide the validation time of the keys and can also stop his parenthood with any of his children at any time.
In [Sta01], authors propose an additional feature to the scheme which takes into account peer-to-peer connections. Indeed, a peer-to-peer connection is considered as the addition of two one-way connections in reverse directions. Moreover, following the tree, all security politics decided by parent nodes should be applied to the connections between children nodes. In fact, this new feature aims at adapting resurrecting duckling to smart dust networks.

As a drawback, parents should be protected from being compromised since they possess many shared keys and they define the security politics of their children. Thus, if a parent node is compromised, its children are also compromised.

3.3.3.5 Demonstrative identification

The approach in [BSSW02] is designed for small, temporary and local ad hoc networks. It supports essentially symmetric key establishment between neighbor nodes. An application could be that a foreign Personal Digital Assistant (PDA) tries to communicate with an unknown local printer in order to securely print some documents. The solution includes two phases: a *pre-authentication phase* which is an identification phase and a *key establishment phase* which does authentication and establishes a shared key between two peers.
The first step should be done either via a physical contact or an infrared channel[2]. Some short information, such as the hashes of the public keys of communicating nodes, is exchanged. Then, the second phase is done within a normal radio channel, and the information previously exchanged is used in authentication. Afterwards, a communication key can be produced by a well known key establishment protocol, such as DH [Res99], Secure Sockets Layer (SSL) [FKK96], etc...
The drawback of the solution is that all nodes need a hardware module (infrared or physical contact module) to accomplish the pre-authentication. And the applications of the mechanism are limited to few scenarios.

3.3.4 Security association establishment

A Security Association (c.f. section 3.3) involves the passing of "secret words" or keys to establish a secure connection between communicating parties. In this section, we present two MANET two-part SA establishment approaches.

3.3.4.1 SA establishment with mobility

In [CHB03], Capkun et al. introduce a solution which makes use of mobility to establish security associations. The protocol ensures that each peer of a SA is certain of the identity and the public key of the other peer.
It is assumed that all nodes own a hardware module (infrared or wire). Two ways exist to establish SAs. One is similar to the solution in section 3.3.3.5, where nodes first exchange some short information using the module and then turn to their radio channel for the rest of the exchanges. The other uses the secure module to exchange all necessary information.
Thus, nodes are able to establish SAs on-the-fly when they meet new neighbors thanks to mobility. Furthermore, friendships can also help. Consider two peers A and B, and suppose another node that is neighbor of both peers and is a friend of one of them say A, then a one-way SA from B to A can be established. Or, if a

[2] The same method can be found in sections 3.3.3.4 and 3.3.4.1.

node is at once both friend and neighbor of A and B, the two-way SA between A and B can be established through the node.
However, the scheme requires the participation of network users who should do physical contacts or put their infrared interfaces face-to-face.

3.3.4.2 SA establishment with routing

In [BEGA02], a SA establishment method using source routing is presented. Assume that an initiator knows the identity of its target but not the public key, the objective of the scheme is that two communicating nodes acquire the public keys of each other. And optionally, an initiator can obtain a symmetric key chosen by the target.
The procedure is done within a secured source routing topology discovery process. The key idea is to employ SUCV addresses (c.f. section 3.3.2.4).
An initiator joins its self-issued public key into its route request message and signs the message with the corresponding private key. The target can then check the signature, and whether initiator's identity is the hash of the initiator's public key. If successful, the public key of the target will be delivered to the initiator within a route reply message, which will later be verified by the initiator.
Optionally, if the communicating nodes want to establish a shared key between them, a secret can be encrypted and transported to the initiator within the Route REPly message (c.f. section 2.2.2.1).
The mechanism works only if the identities of peers are determinable. However, due to the lack of control, this is difficult to be ensured in MANETs. Furthermore, the proposition has inherited the weakness of SUCV scheme in which bogus identities can easily be created.

3.3.5 Summary

In this section, we discussed some representative key and security association management schemes. Many of them are designed for specific scenarios, and it seems impossible to find a universal solution. Each solution has its advantages and drawbacks that are compared in tables 3.3, 3.4 and 3.5.
Key management is an important aspect of MANET security, since it is often the base of the secure routing. Except the key management solutions presented in section 3.3, the following schemes can also be applied to MANET secure routing protocols (refer to the next section for all the secure routing protocols quated here):

Distributed ring signature This kind of schemes [RST01] can provide anonymity to the signers of a threshold signature. We can apply them to solutions such as COCA to protect the signers' identities.

Aggregate signature This signature scheme [BGLS03] can aggregate p signatures on p different messages which are signed by p different users into one single, short signature. Thus, they can help to significantly reduce the header length of ADVSIG, endairA, etc...

Proposition	Origin	Initialization	Dimension	Duration	Refreshment
Password-based key agreement	Spontaneous	Pre-shared password	Small	Short	No
Pebblenets	Planned	Pre-shared password	Large	Long	Yes
Resurrecting Duckling	Spontaneous	None	Small	N/A	Yes
Distributed CA (Zhou)	Planned	Dealer	Not large	Long	Yes (part)
Distributed CA (Luo)	Planned	Dealer	Large	Long	Yes (part)
Self-organized PKI	Spontaneous	History	Not large	Long	No
Demonstrative identification	Spontaneous	PKI or pre-shared key	Small	Short	No
Key predistribution	Planned	Key poor	Large	N/A	No
SUCV address	Spontaneous	None	Large	N/A	No
ID-based cryptography	Planned	PKG	Large	N/A	No

Table 3.3: Key management solutions

Multisignature This signature scheme [IN83, MOR01] is similar to the previous scheme, except that the multiple signatures should be computed on a same message. It can be applied to the protocols where some information needs to be certified by multiple nodes, such as ARAN, or a protocol using multi-hand reputation.

Designated verifier signature This scheme [JSI96, Cha96] can make a signature only verifiable by a unique and specific user. So it can be used in MANETs, for example, to hide routing information, or to keep anonymity of nodes within a voting system.

Once keys are successfully managed, we can use them to secure the ad hoc routing. In the next section, we present some MANET secure routing mechanisms.

Proposition	Characteristic	Objective	Mobility	Scenario
Password-based key agreement	DH-based	One global key	Limited	Internal conference
Pebblenets	Cluster -based	One global key*	Limited	Smart dust
Resurrecting Duckling	Physical contact	Pairwise keys	Limited	PAN Bluetooth
Distributed CA (Zhou)	Threshold cryptography	PKI	N/A	N/A
Distributed CA (Luo)	Threshold cryptography	PKI	High	N/A
Self-organized PKI	Transitive trust	PKI	N/A	N/A
Demonstrative identification	Pre-authentication	Pairwise keys	Limited	Small LAN or PAN
Key predistribution	Key poor	Pairwise keys	Sometimes limited	Sensor network
SUCV address	$IP = h(PK)$	Asymmetric key	N/A	N/A
ID-based cryptography	$PK = h(IP)$	Asymmetric key	N/A	N/A

Table 3.4: Key management solutions (cont.)

3.4 Routing security

In chapter 2, it is shown that unsecured ad hoc routing protocols are vulnerable to numerous attacks. In this section, we present some secure MANET routing protocols. Some are existing protocols reinforced by additional security mechanisms, some are new secure routing protocols suggested in the literature.

Most of the secure routing research efforts have been placed on reactive and proactive protocols. Indeed, a hybrid protocol is often locally proactive and globally reactive, so it is more complicate to secure but existing techniques might be blended and applied to it.

Since reactive protocols seem less weighty, they are thus more studied by researchers. Moreover, the source routing is the mostly studied routing algorithm because of its features (it is particularly true within cooperation reinforcement solutions, see section 3.5). Indeed, it permits source nodes to easily control all the intermediate nodes and the integrity of the routes. Other algorithms, on the contrary, let intermediate nodes decide their next hop nodes, thus they seem more difficult to be secured. In other words, it is a trade-off between the flexibility and the security.

Many secure routing protocols suppose that source and destination nodes trust each other, while any intermediate node could be malicious. We present the secure routing objectives that counteract the effects of malicious nodes in the next section.

Proposition	Authentication	Distributed	Lightweight	Attacker exclusion	Robust	Flexibility
Password-based key agreement	Yes	Yes	Yes	No	Yes	No
Pebblenets	Yes	Hierarchic	Yes	No	No	Yes
Resurrecting Duckling	Yes	Hierarchic	Yes	Partial	No	Yes
Distributed CA (Zhou)	Yes	Partial	No	Yes	Average	Yes
Distributed CA (Luo)	Yes	Yes	Average	Yes	Average	Yes
Self-organized PKI	By friends	Yes	No	Yes	No	Yes
Demonstrative identification	Yes	Yes	Maybe	No	Yes	Yes
Key predistribution	Yes	No	Yes	No	No	Yes
SUCV address	No	Yes	N/A	No	No	Yes
ID-based cryptography	Yes	Possible	Maybe	Maybe	Yes	Yes

Table 3.5: Properties of key management schemes

3.4.1 Design requirement

MANET routing protocols should be distributed, self-organized, and able to adapt to changing topologies. Furthermore, the security objectives to be achieved are as follows:

Availability Routes can be found if they exist.

Correctness Discovered routes are real routes. In other words, every link contained in a route must truly exist.

Safety A route in use contains no attacker, otherwise the routing scheme must be able to tolerate the attackers by some means.

Optimal Routes should be as optimal (short, rapid, less congested, etc...) as possible if the security requirements are already met.

Resource efficient A protocol should be as lightweight as possible both in term of cryptographic overhead and routing overhead.

Punishment of malicious nodes If malicious nodes can be identified, they should be punished, otherwise they have no incentive to stop attacking. As punishments, misbehaving nodes can be definitely excluded.

Stability A secured routing protocol must be self-stable under attacks.

Data delivery Data should be correctly delivered to their destinations with freshness, authentication and integrity. When necessary, the confidentiality is also required.

Concretely, the above security objectives can be translated into the following requirements:

Nodes should be correctly identified and authenticated, including source, destination and the intermediate nodes.

Routing information should be protected in term of integrity and authenticity.

Data needs the authenticity and sometimes the confidentiality.

Satisfying both performance and security requirements in one routing protocol is a challenging task. So, we usually look for security mechanisms with few performance penalties. Only if a network is very security-sensitive, the security is considered an absolute priority.

3.4.2 Secure reactive routing

The most famous reactive ad hoc routing protocols are AODV and DSR (c.f. section 2.2.2). Many security solutions are based on them. For example, ARIADNE in section 3.4.2.1, Secure Routing Protocol (SRP) in section 3.4.2.2 and endairA in section 3.4.2.8 are secure source routing protocols, and Secure AODV (SAODV) in section 3.4.2.5 is an AODV-based secure routing protocol.

3.4.2.1 Ariadne

The secure routing protocol Ariadne, based on pairwise keys shared by communicating nodes (key $K_{S,D}$ denotes a key shared by a source node S and a destination node D) and Timed Efficient Stream Loss-tolerant Authentication (TESLA)[3]

[3]TESLA is a variant of the hash chain technique, which is a method to authenticate a large number of messages while keeping the cryptographic overhead limited.

To be able to use a hash chain, the following steps need to be performed in advance: first, each node n_i chooses a random value s_i; then by hashing s_i $\mathfrak{L}$ times, the node n_i obtains a list of $\mathfrak{L}$ values: $h(s_i)$, $h^2(s_i)$, ..., $h^{\mathfrak{L}}(s_i)$; finally each node signs its last value $h^{\mathfrak{L}}(s_i)$ and broadcasts it in the network.

Any $h^p(s_i)$ $(1 < p < \mathfrak{L})$ is then able to be authenticated by all members of the network since $h^{\mathfrak{L}}(s_i) = h^{\mathfrak{L}-p}(h^p(s_i))$. Moreover, if any $h^q(s_i)$ $(p < q < \mathfrak{L})$ is already authenticated, $h^p(s)$ can be authenticated with only $q-p$ hashing operations because $h^q(s) = h^{q-p}(h^p(s))$. Nodes can thus use their hash values in the reverse order of their generation to guarantee the authentication and the integrity of their messages (often by HMAC [KBC97]). In ad hoc networks, proactive routing protocols tend to use TESLA [HJP02] or its variants because they need to send many regular topology messages.

In comparison with hash chain, TESLA has the advantage of economizing the hash chain elements at the cost of a loose synchronization. With TESLA, a node uses only one chain element to authenticate all messages sent in one time interval, and each element will be disclosed after the expiration of its time interval. LHAP (Lightweight Hop-by-hop Authentication Protocol) [ZXSJ03] and [HPT99, Che97] are examples of the TESLA applications in ad hoc networks.

[PCSJ01, PCJS00], is proposed by Hu et al. in [HPJ02]. According to the authors, the protocol can also easily be adapted to two other schemes, namely shared keys between each pair of nodes and digital signatures.
Suppose that a route $S, I_1, ..., I_i, ..., I_{d-1}, D$ will be discovered in a route discovery process. Then the RREQ initiated by the initiator S should be

$$S \rightarrow * : IP_S, IP_D, id, \tau, h_0, \text{ where } h_0 = h_{K_{S,D}}(IP_S, IP_D, id, \tau)$$

τ is the TESLA time interval at the pessimistic expected arrival time of the request to the target D (clock skew taken into account), and id is a random number. A RREQ rebroadcasted by an intermediate node I_i should be

$$I_i \rightarrow * : IP_S, IP_D, id, \tau, h_i, IP_{I_1}, ..., IP_{I_i}, M_{I_1}, ..., M_{I_i}$$

where $h_i = h(IP_{I_i}|h_{i-1})$ and

$$M_{I_i} = h_{K_{I_{i_\tau}}}(IP_S, IP_D, id, \tau, h_i, IP_{I_1}, ..., IP_{I_i}, M_{I_1}, ..., M_{I_{i-1}})$$

$K_{I_{i_\tau}}$ is the TESLA key of the node I_i at the time interval τ. Upon receiving the RREQ, D will compute an HMAC and send back a RREP:

$$D \rightarrow I_n : IP_D, IP_S, \tau, IP_{I_1}, ..., IP_{I_{d-1}}, M_{I_1}, ..., M_{I_{d-1}}, M_D$$
$$\text{where } M_D = h_{K_{S,D}}(IP_D, IP_S, \tau, IP_{I_1}, ..., IP_{I_{d-1}}, M_{I_1}, ..., M_{I_{d-1}})$$

Each intermediate node I_i will not send back the RREP until $K_{I_{i_\tau}}$ can be revealed. The RREP sent by node I_i to node I_{i-1} should be:

$$I_i \rightarrow I_{i-1} : IP_D, IP_S, \tau, h_{d-1}, IP_{I_1}, ..., IP_{I_{d-1}}, M_{I_1}, ..., M_{I_{d-1}}, M_D, K_{I_{(d-1)_\tau}}, ..., K_{I_{i_\tau}}$$

Node	By receiving the RREQ	By receiving the RREP
S		$K_{I_{i_\tau}}$, M_D, M_{I_i} $(1 < i < d-1)$
I_i	τ is not reached, id	wait for the disclosure of $K_{I_{i_\tau}}$
D	τ is not reached, h_{d-1}	

Table 3.6: Fields to be verified in Ariadne

Fields to be verified at each step of RREQ and RREP handling are summarized in table 3.6.
Ariadne is able to provide authentic routes to initiators, since an initiator can authenticate each hop of a route and no node can be removed from a route. However, the protocol is less adapted to large dimension MANETs. This is because, first, the length of header increases rapidly with the length of route; second, it is difficult to estimate τ in advance when network is large, thus nodes might wait for a long time before sending any traffic even though they are close to each other; third, due to the utilization of TESLA, delay exists also for RERRs, which just needs a rapid reaction to avoid data loss. Ariadne does not cope with wormhole attacks.
In Ariadne, TESLA keys need to be authenticated. For this, there is a solution named ID-based message authentication (IDHC) [Mic04] which can bind the identities of nodes and their TESLA keys together.

3.4.2.2 Secure Routing Protocol (SRP)

Secure Routing Protocol [PH02] aims at providing authentic routes to source nodes with a minimum overhead. SRP can be an extension of any existing reactive protocol especially DSR. It supposes that there is a SA between each pair of communicating nodes, within which shared a secret key $K_{S,D}$, a random seed and the expiry time of the SA. The establishment of the SA is not described in detail in the paper but it is supposed that asymmetric keys and the DH protocol can be used for the purpose.

A SRP header should be added to a reactive routing protocol header. It is composed of six fields including an SN (a sequence number), a id (a random number seeded by the seed in the SA), and a keyed Message Authentication Code (HMAC). When an initiator sends out a RREQ, it computes the HMAC as the hash of the message, by using $K_{S,D}$.

Intermediate nodes are not allowed to reply to the RREQ, but they should check the id and the SN, and append their identities to RREQ before rebroadcasting it.

Upon receiving a RREQ, the destination checks the HMAC calculated by the initiator for the authenticity of the request. Afterwards, it returns a RREP to initiator and protects the whole RREP by another HMAC using $K_{S,D}$. The initiator will check the validity of this new HMAC to ensure the integrity and the authentication of the route in the RREP.

Optionally, we may employ the Intermediate Node Reply Token (INRT) mechanism which uses multi-part SA. That is to say, nodes in a multi-part SA can reply to the requests of each other.

SRP uses a Neighbor Lookup Protocol (NLP) where the binding of IP and MAC addresses can prevent a lot of impersonation attacks. However, since no signature is required, and since MAC addresses are nowadays as easy to be spoofed as IP addresses, it will be difficult for NLP to detect all spoofing attacks, except if there is a conflict or a central server to detect bogus addresses.

In addition, there is a DoS (c.f. section 2.6.4) prevention mechanism in SRP: a node sending too many route requests is dropped in priority. Hence, when there are two requests, the one from a higher priority node will be served before the one from a lower priority node.

However, SRP is simple but not failsafe. First, it cannot efficiently protect its route maintenance phase because no intermediate node is really authenticated. Therefore a malicious node can act correctly in the topology discovery phase but impersonate to send wrong RERRs (thanks to NLP, such an attack may be detected but the attacker can never be identified). Second, it is shown in [Mar02] that the formal proof proposed by SRP is not reliable, thus some incorrect topology information can still be returned to S (c.f. [XLB04] for such examples). Moreover, wormhole attacks can also disrupt SRP.

SRP secures only the topology discovery phase. Thus, it is suggested that SRP works together with another secure routing protocol SMT (c.f. section 3.4.2.3) which secures the data forwarding phase. Most SRP flaws can be covered up by SMT.

3.4.2.3 Secure Message Transmission (SMT)

The Secure Message Transmission (SMT) protocol [PZ03] supposes that a set of reliable paths is provided to each sender by SRP. It uses also the other hypotheses of SRP.
The basic idea of SMT is to break each data message into several small pieces and send them through a set of diverse, preferably node-disjoint paths. A path set is named an *Active Path Set* (APS) that should be a subset of all existing paths from a source node to a destination node. Since some redundancy is introduced into the computation of the pieces, even if not all pieces are received, the original message may be successfully reconstructed at the destination. Then, an acknowledgement will be sent back to the source node. However, when there are not enough pieces to reconstruct the message, the destination node should inform the source node the paths on which pieces have been successfully delivered. Thus, the sender can perform some retransmissions through the operational paths. To guarantee the integrity and the authentication of each piece, all pieces are sent with an HMAC computed with key $K_{S,D}$.
The reliability of the paths in an APS will be evaluated by the sender. A high rating will be given to a path on which pieces are always correctly delivered, whereas failing paths will be chosen less frequently or rejected from the APS due to their low ratings. This mechanism is adaptive to both topology changes and attacks.
We will compare SMT to TRP in chapter 5.

3.4.2.4 Authenticated Routing for Ad hoc Networks (ARAN)

Proposed by Dahill et al., Authenticated Routing for Ad hoc Networks (ARAN) [SDL+02] verifies routes in a hop-by-hop manner. It is neither a distance vector protocol nor a source routing protocol.
ARAN requires a CA server CA and assumes that the public key of the server PK_{CA} is known to all nodes. Each node denoted A has to obtain a certificate $cert_A =< IP_A, PK_A, t, e >_{SK_{CA}}$ before entering the network, where t and e are respectively the creation time and the expire time of $cert_A$, and SK_{CA} is the private key of CA.
At the beginning of a route discovery phase, an initiator S broadcasts a Route Discovery Process (RDP) message: $< IP_D, cert_S, SN, t >_{SK_S}$, where D is the destination node, and SN is a monotonically increasing sequence number. The pair (SN, IP_S) (IP_S can be found in $cert_S$) will be used by intermediate nodes to verify the freshness of RDP.
The first intermediate node I_1 adds its signature and its certificate to the RDP message and then rebroadcasts it: $< < < IP_D, cert_S, SN, t >_{SK_S} >_{SK_{I_1}}, cert_{I_1} >$.
Any following intermediate node I_i $(2 < i < d)$ verifies the signature of I_{i-1}, removes it from the message, then adds its own signature and certificate to the message before rebroadcasting it: $< < < IP_D, cert_S, SN, t >_{SK_S} >_{SK_{I_i}}, cert_{I_i} >$.
The destination D takes the first received RDP (not necessary the shortest route) and replies it with a REP (REPly) message. The REP is sent in a similar way as

RDP but on the reverse path. Upon receiving the REP, S verifies the signatures of D and I_1, and checks whether SN is valid.
RERR messages are signed by their originators. Intermediate nodes have no right to modify them.
If S wants absolutely the shortest route, it can further broadcast a Shortest Path Confirmation (SPC) message:

$$S \rightarrow * :< \{IP_D, cert_S, < IP_D, cert_S, SN, t >_{SK_S}\}_{PK_D} >$$

Any intermediate node should resign it, add cryptographic credentials to it, and reencrypt it with PK_D. The destination D replies to the first SPC, and also any later SPC with a shorter path, after verifying all their signatures. A reply is a Recorded Shortest Path (RSP) message: $< IP_S, cert_D, SN, route >_{SK_S}$. The procedure guarantees that no node can be removed from a route, and any intermediate node is authenticated, thus the shortest route can be found.
In ARAN, a certificate $cert_A$ can be revoked by the server CA with a revoke message $< cert_A >_{SK_{CA}}$. Each node, upon receiving a revoke message, invalidates all routes passing through the revoked node.
ARAN provides the authentication, the non-repudiation and the integrity to its routing control messages, at the price of weighty cryptographic operations and an online CA server. The effects of revocations will depend on the topology of the network, since malicious nodes may not forward revocation messages. ARAN is still vulnerable to wormhole attacks.

3.4.2.5 Secure AODV (SAODV)

Manel Guerrero Zapata et al. suggested a Secure AODV [PBRD02] protocol (SAODV) in [ZA02, GZ02]. SAODV protects two fields, namely SN and HC, in AODV routing messages. HC is the only mutable field that is increased each time the message is rebroadcasted.
SAODV needs a CA server to manage a PKI, with which every routing message can be signed by its originator. Therefore, except HC, the message's integrity and authentication are ensured.
One hash chain is used to protect each HC field. An initiator puts s (a random seed) and $h^{Max\_hop\_count}(s)$ into each RREQ. Each intermediate node increases the HC and replaces the field of s by its hash s'. Then, any node can verify the HC field by checking whether

$$h^{Max\_hop\_count}(s) = h^{Max\_hop\_count-HC}(s')$$

A malicious node is not able to decrease the HC since it does not know the appropriate hash value. However, this mechanism can only prevent nodes from decreasing the HC but is unable to detect the same-distance fraud (a malicious node does not increase the HC). Moreover, when there are some colluding attackers, the route length can still be reduced by communicating an old s or s' value to a downstream accomplice.

The utilization of SN is also restricted in SAODV. In a nutshell, nodes can no more increase their SNs as they wish. For example, no more increasing of SN when a RERR is generated, or a temporary leaving node should keep its SN and reuse it when it comes back. These measures can prevent nodes from deliberately increasing SN thus attracting traffics.
If intermediate nodes are allowed to reply to RREQs, a double-signature called *signature for RREP* should be generated with RREQ and be sent back by a responder within its RREP. The signature provides information that can restrict the RREP, thus the initiator is sure that the responder has received the correct RREQ.
SAODV guarantees the integrity and authentication of AODV routing control messages. However the use of hash chains cannot totally prevent attacks on HC.

3.4.2.6 Security-Aware ad hoc Routing (SAR)

The Security-Aware ad hoc Routing (SAR) protocol [YNK02] is proposed by Yi et al.. It can be considered as a framework which can incorporate any existing routing protocol. In SAR, every node, every application, and even every packet has a *security level*. A node can forward, rebroadcast or reply a packet if and only if it has a security level equal to or higher than the level required by the packet.
One secret key is defined for each security level, and each key should be distributed to all nodes having an equal or higher security level. The contents (including the header) of packets should be encrypted by the key of their corresponding security levels, thus low level nodes cannot read them. Consequently we can also protect topology information from being discovered by unauthorized nodes.
SAR is adapted to networks such as military networks where a general has a higher security level than a soldier. However, the key management in SAR may be complicated if a high level node can be compromised. And the evaluation of security levels should also be considered to make the model more adaptive.

3.4.2.7 Secure Position Aided Ad hoc Routing (SPAAR)

Several ad hoc secure routing protocols rely on GPS, such as the Secure Position Aided Ad hoc Routing (SPAAR) protocol [CY02]. With GPS, nodes can be sure of their geographical positions, therefore neighbor relationships will be much easier to determine.
Every node broadcasts regularly two kinds of messages in its neighborhood: the *public key distribution* message which contains the certificate of the node, and the *Hello* message which contains the current position and the transmission range of the node. The neighborhood can then be calculated.
Every node further generates a pair of *asymmetric neighbor keys* and sends the public key to its authenticated neighbors. All afterward routing control messages including RREQ, RREP and RERR will be signed by both the global private key and the neighbor private key of the sender. Besides, the whole routing discovery process is similar to ARAN.
SPAAR can be used in hostile environments where the security is an important concern. Furthermore, SPAAR can prevent wormhole attacks since it provides a

solid neighbor lookup protocol.

3.4.2.8 endairA

Levente Buttyan and Istvan Vajda proposed in [BV04] a secure protocol named endairA. It supposes that all links are symmetric, and each node has a single and unique identifier. It is equally assumed that although the spoofing attacks are very possible, all nodes are sure of their neighbors, and every node can overhear the communications of its neighbors. But cooperating compromised nodes are not taken into account.
In endairA, nodes add their signatures only to RREP. Then, a RREQ will be

$$< IP_S, IP_D, id, cumulated\_list\_of\_IP\_addresses >$$

and a RREP will be

$$< IP_S, IP_D, id, complete\_list\_of\_IP\_addresses, cumulated\_list\_of\_signatures >$$

The protocol ensures that no incorrect routing information can be accepted by source nodes.
As a summary, the presented reactive protocols are compared in table 3.7 and table 3.8.

Proposition	Routing type	Cryptographic primitives	Synchro-nization	RREP by middle nodes	Others
ARIADNE	SR	$SA_{S,D}$+TESLA	Loose	No	
SRP	SR	$SA_{S,D}$	No	No	
ARAN	Hop-by-hop	PKI	No	No	the fastest route
SAODV	DV	PKI+hash chain	No	Optional	
SAR	All	Symmetric keys	No	Yes	Layered Network
SPAAR	ARAN-like	PKI	No	No	GPS
SMT	SR	$SA_{S,D}$	No	N/A	Multiple routes
endairA	SR	PKI	No	No	Neighborhood

Table 3.7: Secure reactive routing protocols

3.4.3 Secure proactive routing

The operation manner of proactive routing protocols (c.f. section 2.2.2) is very useful when a high routing performance is required, but it is also very challenging

Protocol	Eavesdrop	Spoofing	Greyhole	Blackhole	Wormhole	DoS
ARIADNE	Passive	No	*Active* − $1-ta$	*Active* − $1-ta$	*Active* − $2-ta$	*Active*− $1-ta$
SRP+SMT	No	*Active* − $1-ta$	No	No	*Active* − $2-ta$	No
ARAN	Passive	No	*Active* − $1-ta$	*Active* − $1-ta$	*Active* − $2-ta$	*Active*− $1-ta$
SAODV	Passive	No	*Active* − $1-ta$	*Active* − $1-ta$	*Active* − $2-ta$	*Active*− $1-ta$
SAR	No	Possible	*Active* − $1-ta$	*Active* − $1-ta$	*Active* − $2-ta$	*Active*− $1-ta$
SPAAR	Passive	No	*Active* − $1-ta$	*Active* − $1-ta$	No	*Active*− $1-ta$
endairA	Passive	No	*Active* − $1-ta$	*Active* − $1-ta$	No	*Active*− $1-ta$

Table 3.8: Attack possibilities on secure reactive protocols

when a high level security is necessary, since securing a proactive routing protocol requires continuously securing the whole network topology.
If no compromised node is to be considered, the main security effort should be placed on key management, since proactive messages are regular and simple, and we can simply reject the messages that cannot be correctly authenticated. But, once there are compromised nodes, every entry in every routing message could be incorrect, thus more efforts should be done to prevent attacks from compromised nodes.
Two major proactive routing protocols are OLSR [CJ03] and Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) [PB94]. In section 3.4.3.1, we explain how to secure DSDV. In section 3.4.3.2, we address security issues of IntrAzone Routing Protocol (IARP) [HPS02]. And in section 3.4.3.3, we discuss several secure mechanisms for OLSR.

3.4.3.1 Secure Efficient Ad hoc Distance-vector (SEAD)

In DSDV [PB94], every node has a routing table in which every entry contains essentially three fields: the address of a destination, the *metric* (the known shortest distance to the destination) and the next hop on the shortest route. Topology messages are periodically exchanged between neighbors to keep all the routing tables updated. DSDV-SQ (DSDV for Sequence Numbers), a variant of DSDV, outperforms other DSDV versions by initiating triggered updates upon receiving a SN update.
Secure Efficient Ad hoc Distance-vector (SEAD) [HJP02] is a secure protocol based on DSDV-SQ. In reality, SEAD protects only two fields in topology exchange messages, namely *metric* and SN, from being altered. Note that *metric* is a mutable field.
To attract traffic, an active attacker can either decrease a *metric* or increase a SN.

SEAD can prevent these attacks by using hash chains. For any entry with a SN and a *metric* j, the source node joins a hash chain element $h^{\mathfrak{L}-SN*d\_Max+j}$ to the entry for the authentication of the two fields, where $\mathfrak{L}$ is the length of the hash chain and d_Max is the maximum route length. Since attackers have no knowledge about $h^k(k < \mathfrak{L} - SN * d\_Max + j)$, they are not able to modify the two fields as they like. Furthermore, these two fields can be authenticated immediately by all nodes without any key being disclosed.
However, SEAD does not address other DSDV attacks, and we believe that in SEAD hash chains need to be regenerated frequently since they are rapidly consumed.

3.4.3.2 Secure Link State routing Protocol (SLSP)

Papadimitratos and Haas proposed the Secure Link State routing Protocol (SLSP) in [PH03]. SLSP can either be used as a stand-alone protocol or be used as IARP [HPS02]. It assumes that each node has a pair of asymmetric keys and there are only individual attackers.
SLSP is composed of four components, they are respectively a Neighbor Lookup Protocol, a Public Key Distribution (PKD) protocol, a LSU (Link State Update) exchange protocol and a basic DoS attack prevention mechanism. Among them, the NLP and the DoS prevention mechanism are the same to that of SRP (c.f. section 3.4.2.2).
Each node periodically broadcasts within a *zone* (a zone can be represented by a certain number of hops) a PKD packet that contains the certified public key of the node. The LSU messages are signed and periodically broadcasted within the same zone. The hash chain technique is used to control the TTL field in both PKD and LSU messages.
In SLSP, the "same distance fraud" attack is also possible due to the use of hash chain. Furthermore, the protocol does not really take into account compromised nodes or colluding attackers who can either forge links or initiate wrong *metrics*. However, those problems can be resolved with a high performance sacrifice, see the next subsection for an example.

3.4.3.3 Advanced signature (ADVSIG)

In [RACM04], Raffo et al. proposed an **ADVanced SIGnature** (ADVSIG) system to reinforce the security in OLSR (c.f. section 2.2.2.2). Two hypotheses are used: nodes are synchronized, and a PKI has been established which ensures that all public keys are known to all nodes. ADVSIG messages are added in conjunction with both *HELLO* and *Topology Control* (TC).
Each ADVSIG message contains some so-called *certificates* and *proofs*. A *certificate*, only carried by HELLO messages, contains a neighbor's address, the link state with the neighbor, the timestamp of the message creation and a signature (signed by the initiator). And a *proof*, carried by both HELLO and TC messages, contains the address of the node which initiated the message, the link state with a neighbor that can be extrapolated, the timestamp of the proof creation and a signature (signed by the neighbor).

The basic idea of ADVSIG is as follows. Let each node store some recent atomic neighbor information signed by its neighbors – proofs. When sending a HELLO message at time interval τ_i, a node must join the proofs provided by its neighbors at time interval τ_{i-1} to prove its link state with them at time interval τ_{i-1}. Since a link state at time interval τ_i depends on the link state at time interval τ_{i-1}, the receiver of the message can check the coherence and find abnormalities. With regard to a TC message, each MPR selector information should be confirmed by the MPR selector with proofs. Moreover, a global timestamp and a global signature (signed by the initiator) calculated on the ADVSIG message and the corresponding OLSR message is added to each ADVSIG message.
The ADVSIG mechanism is designed to prevent malicious nodes from inventing inexistent neighbors, under condition that there is no colluding compromised node. The attacks such as replay, message alteration, are precluded since messages are timestamped and signed. To reduce the storage consumption and to limit overhead, authors have suggested to use either 128-bit RSA or 320-bit DSA signatures [ACL+05]. A more detailed analysis of ADVSIG is presented in section 6.3.

See table 3.9 and table 3.10 for a summary of the solutions presented in this section.

	ADVSIG+OLSR	SLSP	SEAD
Type	Link state	Link state	Distance Vector
Cypto.Primitives.	PKI	PKI	Pairwise keys or PKI
Synchronization	Yes	No	Loose synchro if TESLA
Hash Chain	No	Yes	Yes

Table 3.9: Secure proactive routing protocols

	ADVSIG+OLSR	SLSP	SEAD
Eavesdrop	Passive	Passive	Passive
DoS on TC/LSU	No	No	Active-1-ta
Modification of TC/LSU	No	No	No
Masquerade	No	No	No
Gray Hole	Active-1-ta	Active-1-ta	Active-1-ta
Black Hole	Active-1-ta	Active-1-ta	Active-1-ta
Wormhole	Possible to be prevented	Active-2-ta	Active-2-ta
Other link inventions	No	Active-1-ta	Active-1-ta
Same-distance fraud	No	Active-0-1	Active-0-1

Table 3.10: Attack possibilities on secure proactive protocols

3.4.4 Mechanisms against some specific routing attacks

Some routing attacks have attracted special attentions, either because they are particularly difficult to prevent, such as wormhole or byzantine attacks, or because

they are newly found attacks that existing secure routing protocols have not taken into account, for example rushing attacks. We detail in the following some security mechanisms against these sophisticated attacks, which usually require many hypotheses.

3.4.4.1 Wormhole attack

Hu et al. has designed **packet leash** as a solution to protect routing protocols against wormhole attacks (c.f. section 2.5) [HPJ03]. A *leash* is defined as any information appended to a packet to restrict the maximum transmission distance of the packet. Two kinds of leashes are proposed: *geographical leash* and *temporal leash* (where faster wormhole attacks are not considered).
Geographical leash depends on the GPS system and a loose time synchronization. That is to say, each node n_i permanently knows its position P_i. Also, it is assumed that nodes have a maximum moving speed v.
Suppose that a packet sent by node n_1 at time t_1 is received by node n_2 at time t_2. n_2 measures the distance dst between n_1 and itself at time t_2 with information provided by GPS, then verifies whether dst is smaller than $dst_{t_1} + 2 * v * (t_2 - t_1 + \triangle) + \partial$, where $\triangle$ is the corrective value for relative time errors and ∂ is the corrective value for relative distance errors. Any authentication technique can be used for this scheme.
Temporal leash relies on tight time synchronization, and the maximum time error $\triangle$ (1 μ s in tests) should be known to all nodes. Moreover, each transmitted packet has an expiration time $t_e = t_1 + L/c - \triangle$, where L is the transmission range, c is the radio propagation velocity (speed of light) and t_1 is the packet sending time. The receiver n_2 checks if the packet receiving time t_2 is not after t_e. If $t_2 \leq t_e$, the packet is considered as free of wormhole attacks. Pairwise symmetric keys can be used for the authentication in this scheme.
A more efficient authentication scheme, **TESLA with Instant Key disclosure (TIK)**, is further proposed for temporal leashes. Unlike TESLA, TIK permits instantaneous authentication of messages, since a TIK key TIK_i can be disclosed in the same packet that is going to be authenticated. A TIK packet is of the form: $< h_{TIK_i}(M), M, MHT, TIK_i >$, where M is the message and MHT is the merkle hash tree information (the *Merkle Hash Tree* (MHT) model [Mer80] is used to facilitate the authentication of disordered hash values). A sender should guarantee that, when TIK_i is sending out, the disclosure time t_i of TIK_i is reached; and, the time that the end of HMAC is received by the receiver will be earlier than $t_i - \triangle$. If these two conditions are satisfied, TIK_i will be a valid TESLA key, and the authenticity of the message can be determined immediately without any message buffered. However, TIK cannot be used if the maximum transmission range is smaller than $c * \triangle$, and TIK packets should have a minimum length.
The solutions can help receivers to determine if a received packet is sent by a neighbor or by a wormhole attacker. However, packet leashes require that GPS and synchronization mechanisms are reliable and can provide unalterable position and time information. Furthermore, a receiver is supposed to be a benign node, which is not pertinent since wormhole attacks are often committed by colluding attackers.

Proposed by Capkun et al., the **SECure Tracking Of node encounteRs (SECTOR)** protocol [CBH03] presents also a countermeasure against wormhole attacks. More generally, SECTOR allows nodes to prove their encounters with other nodes. Several hypotheses are assumed: a loose time synchronization; nodes are able to measure their local timing with a nanosecond precision; the pre-establishment of security associations between each two nodes; a central authority that controls the network membership; unique identity for each node; and a special module that can temporally take over the control of the radio transceiver unit from the CPU. It is supposed that when the module takes control of a node, the node can reply a one-bit response to a one-bit request without any delay, congestion or jamming.
Two protocols are designed to realize SECTOR: a *Mutual Authentication with Distance-bounding (MAD) protocol* and a *Guaranteed Time of Encounter (GTE) protocol*. MAD is in charge of authentication and distance determination, while the GTE protocol proves the accurate time of encounters.
MAD is indeed the *Brands and Chacum protocol* [HPJ03] with slight modifications. The basic idea is, two nodes exchange a series of challenges and responses as soon as possible. Then the average time between the challenges and responses is used to compute the distance between the two nodes. MAD uses symmetric primitives and hash operations to ensure the mutual authentication.
GTE provides "proofs" that one node encountered another node at a given time. The mechanism is based on MHT: each node has a MHT tree and every leaf of the tree contains different time information. To prove to a node C that node A and B had encountered at time t, A and B should, once encountered at time t, exchange messages according to the MAD protocol, then exchange their MHT leaves containing the time information t. Then those leaves can later be verified by C as proofs.

3.4.4.2 Byzantine attacks (blackhole attack)

Awerbuch et al. proposed a secure routing protocol resistant to byzantine attacks (c.f. section 2.4) [AHNRR02]. Three kinds of Byzantine attacks are considered: loop, non-optimal route and blackhole/greyhole. The routing is the source routing and is authenticated hop-by-hop by asymmetric keys.
The basic idea is to identify *faulty links* rather than attackers. To achieve the goal, every link is rated. Depending on the ratings of the links on a route, nodes decide to use the route or not. No node can be directly excluded.
Every data packet should be acknowledged by its destination. If a source node observes that the number of packets unacked violates a threshold, it starts a *fault detection procedure* to determine the faulty link (where the blackhole/greyhole attack happens).
By using the dichotomize method, the source sends some *probe* messages to some of the nodes on the route. All probe messages should be acknowledged until the faulty link can be localized. Then the sender decreases the rating of the faulty link. For the security of the procedure, a pairwise key is supposed to exist between the source

and each probed node.
However, in theory probe messages should be indistinguishable from normal data packets. Otherwise, attackers can reply only to probes while still dropping data packets.
Another mechanism to counter blackhole attacks is proposed in [HWD02] by Deng et al.. Its basic routing protocol is AODV, where intermediate nodes are allowed to reply to RREQs, and any intermediate node who replies to a RREQ should also add its next hop to RREP.
The scheme is based on redundancy topology discovery messages. Suppose that a RREP initiated by an intermediate node I_i reaches its source node S, then a supplement RREQ *FurtherRouteRequest* will be broadcasted by S to the next node I_{i+1} indicated in the RREP. The second RREQ helps to test whether a route to I_{i+1} really exists. I_{i+1} will return a *FurtherRouteReply* message upon receiving the *FurtherRouteRequest* within which it should tell S whether it really has a route to the destination. However, the solution is not able to cope with cooperating attackers, and blackhole attacks in the data forwarding phase are not really considered.

3.4.4.3 Rushing attack

In [HPB03], Hu et al. developed the *Rushing Attack Prevention* (RAP) protocol as a defense to rushing attacks (c.f. section 2.6.2.1). The protocol can be combined with AODV, DSR or some other secure reactive routing protocols such as Ariadne. RAP assumes that the network is always connected and most links are bidirectional. It is also assumed a loose synchronization and the existence of a public key server. An *instantly-verifiable broadcast authentication protocol*, and a wormhole attack preventer (TIK, packet leashes, etc...) are also required.
In RAP, the first received RREQ will not be systematically rebroadcasted. Instead, each intermediate node should gather p RREQs from p different neighbors and randomly choose one RREQ to rebroadcast. Otherwise, the rebroadcast will be done after a timeout if a node fails to gather p RREQs.
The *Secure Neighbor Detection* (SND) technique is used to reject all unidirectional links from communication. It is combined with the routing protocol in such a way that after receiving a RREQ, a node should check its neighborhood at real-time and obtain a *Route Delegation* message from the upstream node before rebroadcasting the RREQ.

3.4.4.4 Sybil attack

Sybil attacks (c.f. section 2.5) are not easy to be detected or countered when there is no central server. To react against sybil attacks, Newsome et al. proposed some defenses in [NSSP04].
The first approach *Radio Resource Testing* supposes that each node has only one radio module. Thus, nodes cannot send or receive simultaneously on more than one channel. A node (applicant) can then check if there are sybil identities in its neighborhood by asking each of its "neighbors" to send it a message on a certain channel. The applicant chooses randomly a channel to detect whether all messages

on the channel are sent. A "neighbor" who does not send the message is considered as a sybil identity. However, this solution cannot tell the real identity of the attackers, and multiple tests must be performed to obtain a good detection probability.
The second approach depends on a special random key predistribution scheme (c.f. section 3.3.3.3) with which keys are distributed to nodes according to their identities. It is supposed that a node can present only one identity to the key pool, thus any node is not able to obtain more than one key set. Therefore, all nodes know which keys should be owned by which node. Thus, they can send challenges to others to test if other nodes really know what they should know. If a test fails, it is very likely that there is a sybil identity. Again, with this solution we can only detect sybil identities but will not be able to identify the sybil attackers.

3.4.5 Summary

In this section, we presented the main mechanisms proposed for the secure routing of MANETs, and also the drawbacks of these mechanisms. Even though diverse methods are employed, they are always based on some key schemes, such as, thanks to a key infrastructure, ad hoc routers can be authenticated and routes can then be established only among authorized nodes; techniques as TESLA or hash chain can provide lightweight methods to protect sensible *metrics* from being altered; and so on. Moreover, several special modules like GPS can offer important information to nodes, in such a way that even some sophisticated attacks may be prevented.
The mechanisms presented in this section mainly secure the routing discovery phase from malicious attacks. In the next section, we introduce some cooperation reinforcement mechanisms, which mainly counter the selfish behaviors in the data forwarding phase.

3.5 Cooperation reinforcement

Most of the mechanisms presented in the previous section do not allow MANETs to combat selfish nodes. This problem is addressed in this section.
Four types of solutions are discussed in this part. In subsection 3.5.3, several reputation-based solutions are introduced. In subsection 3.5.4, we describe one token-based mechanism. A solution based only on first-hand supervision (without reputation system) is in subsection 3.5.5, and we present some incentive methods using micro-payment in subsection 3.5.6. The section is summerized in subsection 3.5.7.

3.5.1 Design requirement

An efficient cooperation reinforcement scheme should have the following properties:

Stimulate cooperation It should encourage nodes to cooperate in routing.

Lightweight It should be lightweight and have low influence on network performance.

Punish selfish node Selfish nodes that refuse to cooperate should be punished. They can be temporarily or definitively excluded from the network, or be prevented from sending and receiving until they become normal.

Distributed It should be distributed, thus difficult to be attacked at a central point.

Robust to attack It should introduce as few flaws as possible (for example, the possibility of the blackmail attacks). Also, it should remain operational even under attacks.

Authentication It is indispensable that messages and nodes are correctly authenticated, and judgments are correctly attributed.

3.5.2 Selfish node model

Three selfish models are studied in [Mic04]. They are respectively *selfish forwarding model* (model I), *selfish routing model* (model II) and *energy-driven selfish behavior model* (model III).
With model I, nodes systematically refuse to forward data packets while still participate in the topology discovery phase. Thus, nodes can largely extend their battery lifetimes. This behavior is very harmful since it can cause a low packet delivery ratio. On the other hand, it is also a blackhole attack (c.f. section 2.6.3). This model works with all the ad hoc routing protocols. It must be prevented.
Remark: We can refine the model by adding a sub-model *selfish partial-forwarding* (I.I). It can simulate the greyhole attacks with which packets are partially dropped.

The *selfish routing model* (model II) describes the behavior of the nodes that participate in neither the topology discovery phase nor the data forwarding phase. In other words, these nodes will never be intermediate nodes, instead they will only be sender or receiver. This behavior is very useful for saving energy but less dangerous in term of security, since only the availability of the network service is harmed.
Remark: It should be noted that this model cannot be directly applied to any MANET routing protocol. To our opinion, reactive protocols allow in particular the selfish routing model. This is due to the fact that, with a reactive protocol, a node adopting such behavior can prevent itself from being included in the routing while still be able to send packets and receive (it is sufficient for it to reply to any RREQ destined to it).
On the contrary, with a proactive protocol, to receive data or to be included in the routing tables of the other nodes (in other words, to be known to the other nodes), any node should at least partially participate in the topology information exchange. Therefore, we could say that this model cannot work with proactive protocols.
Take as example the OLSR protocol, a node which sends neither Hello nor TC will be excluded from the network since no node can establish a link with it. Thus data destined to it cannot be sent out due to the lack of route. Nevertheless, the node can still be a source node if it receives from the network the routing information.

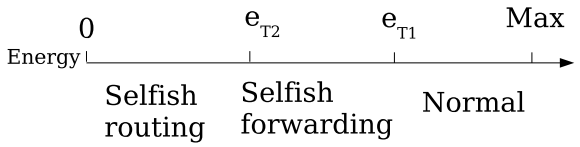


Figure 3.1: Energy-driven selfish model

Otherwise, OLSR also gives another opportunity to selfish nodes. That is to say, a node which sends only Hello but not TC will stay in the network but never become an intermediate node. Thus the data delivery ratio of the network can be decreased if it is chosen as MPR.

The *energy-driven selfish behavior model* (model III) combines the two former models and tries to provide a psychological explication to the selfish behaviors. Its main idea is that "selfish" nodes adapt their routing behaviors to their energy levels. Three states and two energy thresholds e_1 and e_2 ($0 < e_2 < e_1$) are set in the model, as shown in figure 3.1. When a node has its energy level higher than e_1, it behaves like a normal node; when its energy level is lower than e_1 but higher than e_2, the node performs the selfish-forwarding as in model I; and when the node has its energy level lower than e_2, it follows the selfish-routing as in model II. It is also supposed that once a node has run out of its energy, it will be recharged to the maximum energy level within a time interval. This model may be more realistic when the energy is the only factor of selfishness.

3.5.3 Reputation-based mechanisms

Reputations are the records of person's or agent's actions and the opinions of others about those actions. Reputations can be published in order to allow other people (or agent) to make informed decisions about whether to trust that person or not. A reputation system which uses pre-programmed criteria for reputation management automates the process of encouraging cooperation behavior over selfish behavior.
Most Internet sites which mediate between large numbers of people use some form of reputation mechanism: Slashdot, eBay, ePinions, Amazon, and Google all make use of collaborative filtering, recommender systems, or shared judgments of quality. Reputation systems are well adapted to the distributed systems without a-priori trust. Therefore they can be well adapted to ad hoc networks. They can provide rankings to nodes, thus distinguish misbehaving nodes from benign nodes. However, they are not suitable for ephemeral MANETs since in ad hoc networks, post-priori reputations need some time to be established.
Many reputation systems collect information of routing behaviors with supervision (watchdog). There exist two supervision modes: one is the *supervision on route*, where nodes supervise only packets treated by them; the other is the *supervision in the neighborhood*, where nodes check not only the packets treated by them, but also all heard packets. The first mode has less overhead, while the second mode provides more first-hand information to nodes at the cost of more storage and handling

requirements.
In [MGLB00], drawbacks of watchdog are discussed. They are respectively *ambiguous collision* and *receiver collision*. "Ambiguous collision" depicts a collision at a supervisor, when a node sends a packet to the supervisor the same time the supervised node forwards another packet. "Receiver collision" depicts a collision at the receiver of a packet supervised. Since the supervisor cannot observe the collision, the eventual retransmissions could be considered as replay[4], and no retransmission could be considered as a benign behavior. Then, we can see that supervision is not 100% reliable.
We consider two kinds of reputation systems, according to whether or not they use indirect reputation information. Indirect reputation information is also called second-hand information, recommendations, etc... Indeed, first-hand information is credible, but second-hand, third-hand, etc. information is doubtable to be used directly, even though it can allow us to draw early conclusions about nodes that we have never encountered.
Reputation systems can also be distinguished according to whether or not they consider negative experiences. Indeed, some reputation systems do not consider negative experiences due to the fear of blackmail attacks (c.f. section 2.4). However, to our opinion, this measure reduces the speed of reputation establishment, and it has no substantial difference with the other reputation systems, since positive experiences can also be forged.
In this section, all propositions use source routing as their underlying routing algorithm, because watchdog requires that a route in use is predictable. Other routing protocols are more dynamic but less adapted to a supervision system. This is due to the fact that, since intermediate nodes have the right to decide the next node on-the-fly, a supervising node cannot be sure that the packet is correctly forwarded to the next node.

3.5.3.1 COllaborative REputation (CORE)

The COllaborative REputation (CORE) mechanism to reinforce node cooperation in mobile ad hoc networks [MM02, Mic04] is proposed by Michiardi and Molva. It is designed to fight energy-driven selfish behaviors which are considered by the authors as the most rational selfish behaviors. In CORE, selfish nodes are not excluded but discouraged to be selfish, since benign nodes will not forward their data packets until they become cooperative. The use of second-hand reputations is optional.
In CORE, node identities are supposed to be unique, unspoofable and unforgeable. All nodes are able to perform the promiscuous mode, and the topology discovery phase is already secured. Network traffic is supposed to be dense.
Each node has four CORE components. Two *monitor components* supervise respectively the packet forwarding operations (called function $f = PF$) and the routing discovery operations (called function $f = R$). The monitoring results are then sent to a *reputation manager component* which manages a reputation table. The table

[4] If RTS/CTS is used with CSMA/CA at the MAC layer [Soc05], the collision possibility can be reduced.

maintains the reputation towards each of the other nodes. Finally, the *punishment component* will decide whether or not to penalize the other nodes. CORE is totally distributed and exchanges of reputations inter-nodes are optional.
To perform the monitoring, nodes stock several essential information of the packets passing by (the supervision on route). Every entry is of the form $<$ UID, IP_S, IP_D, MAC_S, MAC_D, h(payload) $>$, where UID is the *unique packet identifier*, IP_S, IP_D, MAC_S and MAC_D are respectively IP and MAC addresses of the sender S and the receiver D, and $h(payload)$ (160 bits) is the hash value of the data payload. In all, each entry occupies 332 bits of storage. To further reduce the storage requirement, the packet supervision rate can be dropped to 20%.
Each packet is identified by three fields: $< UID, IP_S, IP_D >$. CORE compares a packet heard by the promiscuous mode to what it is expecting. In case of modification of data payload or non-forwarding, the reputation on the downstream node decreases, otherwise it increases. To limit the storage consumption and to be adaptive to the selfish model III, only the last V observations on each node are taken into account in the calculation of reputation. The following formula is used by node A to calculate a reputation on node B at time t:

$$r_A^t(B) = \sum_{f \in \{PF,R\}} w_f \{ r_A^t(B|f) + \sum_{z \in \mathcal{N}_A} \lambda_z r_z^t(B|f) \}$$

where w_f denotes the weight of the function f and $\mathcal{N}_A$ is the direct neighbors of node A. If a reputation is locally generated (without recommendation), $r_z^t(B|f)$ equals to 0. Otherwise, $\lambda_z = r_A^t(z|f)$ denotes the weight on the indirect reputation $r_z^t(B|f)$.
CORE is validated by both simulations and the *game theory modeling* [FT91]. It is proved that selfish behaviors will be given up and energy of benign nodes could thus be saved.
However, CORE has several drawbacks. First, the way it saves packets does not permit us to detect all attacks, so we need to refine the scheme. Second, nodes punish selfish nodes by rejecting their packets, but this behavior is itself the same as a selfish behavior (with a different objective but nevertheless the same method), a lot of data loss may be caused due to punishment.

3.5.3.2 Cooperation Of Nodes - Fairness In Dynamic Ad-hoc NeTworks (CONFIDANT)

In [BB02b], Buchegger et al. proposed Cooperation Of Nodes Fairness In Dynamic Ad-hoc NeTworks (CONFIDANT). It uses a PGP-like [S.G95] self-organized PKI scheme [HBC01] (c.f. section 3.3.2.2) as its key scheme. Like CORE, it also supervises the two routing operations: route discovery and data forwarding, in order to detect both selfish and malicious nodes. Unlike CORE, with CONFIDANT a node supervises all its neighbors and uses recommendations. As punishment, nodes refuse to forward RREQs sent by misbehaving nodes. The main goal of CONFIDANT is to establish a correct reputation system as rapid as possible.

CONFIDANT has two versions. In the first one, nodes have four trust levels just as in PGP: *friend*, *marginal*, *unknown* and *enemy*. Each node trusts its friends and records them in a *friend list*. If a node A could identify a misbehaving node X, A will send to its friends a signed ALARM which accuses X. And, if there is further a traffic which is getting through X, A also sends the ALARM to the sender of the traffic. Otherwise, an ALARM can also be broadcasted, and all other nodes decide whether or not to take it into consideration according to their local reputation on the sender of the ALARM.
Before sending any traffic, nodes should choose a "secure" route from its route cache. The most useful criterion to choose the routes and to stimulate good behaviors might be using the nodes with the highest reputations which are calculated based on some most recent behaviors of the nodes.
In the second version [BB03], to avoid possible blackmail attacks, CONFIDANT further uses a Bayesian approach to administrate exchanges of reputations. The approach helps us to tell trustworthy reputations from lies, thus stops liars from spreading wrong accusations. For this, a new parameter, the credibility of nodes, is introduced.
However, it should be noted that the authentication is only used on route discovery messages and ALARM messages, thus spoofing in the data forwarding phase is always possible.

3.5.4 Token-based cooperation reinforcement

The threshold cryptography can also be used to reinforce the cooperation. Suppose an N-node dense network where each node has at least φ neighbors, Yang et al. proposed in [YML02] a token-based approach which uses a (N, φ) threshold cryptography. The basic idea is that all nodes observe all behaviors of their neighbors, from which they can collectively decide whether or not to allow a node to participate in the network by giving or not a token to it. The basic routing scheme is AODV.
To be able to stay legally in the network, each node should obtain a signed token which requires the cooperation of at least φ other nodes. A token is composed of three fields: $< owner\_identity, signing\_time, e >$, where the e is the expiration time of the token which could be late if the applicant behaved well during a long time in the near past. Before the expiration time of a token, its owner should apply a new token to its neighbors.
The proposition is totally distributed. Four interacting components are designed for each node, they are *neighbor verification*, *security enhanced routing protocol*, *neighbor monitoring* and *intrusion reaction*. The intrusion reaction component is in charge of reporting the existence of non-cooperating nodes and of revoking collectively their tokens by using the threshold cryptography. The neighbor verification module checks the legitimacy of neighbors (whether or not they have a valid token) so that only legal neighbors can participate in the AODV routing. The neighbor monitoring module is the watchdog module, and the security enhanced routing is a routing based only on the trustworthy nodes. Malicious nodes are isolated if their tokens are revoked or if they cannot renew their tokens before the expiration time

of the tokens. They are also withdrawn from routing tables of benign nodes.
However, with the localization of services, it is necessary that every node has at least φ neighbors, otherwise token renew requests should be broadcasted to 2-hop neighbors or more. If it is the case, non-localized services will create a loop between routing and security. Furthermore, in order to limit the cryptographic overhead, long term tokens should be given to nodes, which requires that nodes should be able to observe neighbors for a long time, thus a strong mobility may not be appropriate to this scheme.

3.5.5 Cooperation reinforcement with first-hand experience

According to Jiang et al. in [JAA04], the main security problems of MANETs can be solved by letting intermediate nodes reroute packets. That is to say, when a downstream node on a route is found malicious or selfish, the route can be redirected. With such a mechanism, countermeasures can be taken much faster than using whatever reputation system.
The proposition is based on DSR, and the physical layer protocol is supposed to be 802.11. It is also assumed that nodes have a perfect knowledge of their neighbors, and node identities are unspoofable and unforgeable. It is decided also that selfish nodes will only be temporarily excluded from packet forwarding, since it is possible that a selfish-like behavior is due to a network congestion, mobility, etc...
The considered selfish model is the selfish forwarding model. Once a node X is identified by another node B as a selfish node, B purges from its route cache all routes that contain node X as an intermediate node, then B broadcasts (with $TTL = 1$) a Route Redirect (RRDIR) packet to inform its neighbors that X should be bypassed and the route in use is going to be changed. Finally, B reroutes all the following packets either through an alternative route or by broadcasting a new RREQ for the destination. If no route can be found, B should send back a RERR to the sender of the traffic.
However, it is possible that it is the rerouting node B which acts maliciously. To address this issue, the authors have listed some possible attacks making use of the security mechanism itself. For example, B mounts a DoS attack which deliberately forwards packets to X even though it knows that X is selfish; or B directly reports a RERR to the source node without any rerouting attempt. But it seems that all these attacks can be detected by part of neighbors of B, thus the misbehaviors of node B can be detected.

3.5.6 Micro-payment mechanisms

With some cooperation reinforcement solutions, a service demander must "buy" services from its service providers. Therefore, to pay for the services that he needs, a service demander must also provide services to others to earn "money". A cooperation relationship can thus be established.
For MANETs, a service demander or a service provider can be a node, and the service can be the packet forwarding function. If a node forwards packets for other

nodes, it can earn "money". On the contrary, if a node uses other nodes to forward packets, it should pay "money" to those nodes.
Then, a node run out of "money" can be considered selfish since it has not sufficiently participated into the packet forwarding function as an intermediate node. As a consequence, it has no more right to be a sender or a receiver.
Normally, the "money" here indicates a virtual currency. However, an alternative choice provided by some mechanisms permits nodes to pay with real money.

3.5.6.1 Nuglet

In [BH01], Buttyan et al. proposed a cooperation reinforcement solution called *Nuglet*. Indeed, nuglet is a virtual currency, and a beneficiary of network services (a source node or a destination node) should pay nuglets to its service providers (intermediate nodes which forward packets for it). This proposition tries to encourage nodes to forward packets for others, and discourage nodes from flooding the network. The protocol relies on a tamper-resistant device to protect nuglets and the exchanges of nuglets, thus we suppose that there is no attack on nuglets.
Two models are described by this proposal. The first one is called *Packet Purse Model*. With this model, a packet sender has to join sufficient nuglets into a packet before sending it. Then, every intermediate node takes some nuglets from the packet when forwarding it. When there is not enough nuglets left in the packet, the packet will be unfortunately rejected. This model guarantees that a malicious node cannot flood the network, since every flooding will cost it some nuglets.
The main drawback of the model is that the sender should know in advance the number of nuglets necessary for the sending of the packet to the destination. If the number is overestimated, some of the nuglets will be wasted; *Per contra*, the packet will be rejected, and then the nuglets are also lost.
The second model is called the *Packet Trade Model*. With this model, destination nodes must pay in order to receive packets. Each intermediate node purchases with nuglets the packets sent by its upstream node, and resales them more expensive to its downstream node.
The main disadvantage of this design is that the senders are allowed to flood the network. Moreover, to waste nuglets of a destination, malicious nodes can deliberately redirect packets destined to it on longer routes.
However, in both models the mobility is not well taken into consideration. If any intermediate node or a destination node left the network or just changed the location, in the first model, the sender will pay for nothing while in the second model, the last node that forwards the packet will lose some nuglets.

3.5.6.2 Sprite

In [ZCY03], Zhong et al. proposed an approach called *Sprite* which relies on a hybrid network architecture. That is, apart from an ad hoc network, there is another network which contains a trusted *Credit Clearance Service* (CCS) provider.
Each node has multiple network interfaces in the way that it is able to switch from the ad hoc network to the other network and make fast connections to the CCS

server. The authors assume also the existence of a PKI and a secure source routing protocol.
To be able to initiate packets, nodes need to pay credit to intermediate nodes. However, unlike Nuglet, nodes do not directly exchange credit among them. Instead, all credit exchanges pass through CSS.
To gain credit, nodes could either pay real money to CCS, or relay packets for others. In the later case, they should keep a *receipt* each time they forward a packet and report them to CSS. Then, it is up to CSS to determine how to charge source nodes and how to give credit to forwarders. A *receipt* is a small message derived from the content of a packet and signed by the initiator of the packet. It can be considered as a proof of forwarding.
Note that a cost incurs when a node connects with CCS. So, the minimum credit must be greater than the cost of one connection. Also, to discourage colluding selfish nodes to gain credit, amounts of credit and debits given to each node are carefully studied. In particular, CCS overcharges source nodes to make flooding attacks unattractive. And for each transaction, more debits are taken away from the source node than the credit given to the intermediate nodes. Finally, to compensate the loss of credit, CCS periodically returns excess credit to all nodes of the network.
The solution does not need any tamper-proof hardware, but it does need a central server and an additional network interface per node.

3.5.7 Summary

In this section we discussed some cooperation reinforcement mechanisms for MANETs. They are either supervision-based or micro-payment-based. Meanwhile, all of them are reactive solutions.
The supervision-based propositions are compared in table 3.11.

	Selfish model	Routing protocol	Second-hand information	Blackmail attack	Supervision mode
CORE	III	DSR	Optional	Possible	Route
CONFIDANT	I&II	DSR	Yes	Possible	Neighbor
Token-based routing	I&II	AODV	Yes	Possible	Neighbor
Routing based on first-hand experience	I	DSR	No	No	Neighbor

Table 3.11: Comparison of reputation-based solutions

3.6 Conclusion

Throughout the chapter, we have reviewed some important propositions for the routing security of mobile ad hoc networks. We presented them within three axis:

key management, secure routing, and cooperation reinforcement.
The first axis, the key management, is a basic security issue of MANETs, because it provides cryptographic primitives and trust relationships to all other MANET secure routing mechanisms. However, due to the self-organization and the self-configuration of MANETs, the key management mechanisms should be distributed and self-organized. Thus they are different to the mechanisms used in traditional networks. The existing key management mechanisms that are presented in this chapter success in adopting the asymmetric and symmetric cryptographies to MANETs, some by using original key management schemes such as threshold cryptography and SUCV addresses. However, researchers are still troubled by the ultimate issue which is building a key scheme from scratch without infrastructure nor a-priori trust.
The second axis, the secure routing, is essentially the application of cryptographic primitives to ad hoc routing messages to ensure the authentication and integrity of the latter. The different choices of cryptographic primitives, and the different ways to employ them in different routing protocols permit the creation of many different secure ad hoc routing protocols.
The cryptographic operations allow the protocols to authenticate senders, receivers and intermediate nodes, to protect the integrity of routing information, and to prevent external nodes from entering the network. Thus, these secure routing protocols can prevent external attacks as well as many internal attacks, even though they are generally unable to detect some of the more sophisticated attacks, such as wormhole attacks or selfish behaviors. However, many of the solutions are at the price of significant computational and routing overhead, such as ADVSIG and Sprite. This is undesirable for the ad hoc networks with limited bandwidth and processing power. Lightweight protocols like SRP are more likely to be expected by these networks.
The third axis, the cooperation reinforcement, is a new issue presented in the self-organized networks such as peer-to-peer networks. The issue is more serious for ad hoc networks since the nodes in MANETs may be ressource-restained thus they have more intentions to be selfish. The problem can usually be resolved by employing a reputation system which rates nodes according to their routing behaviors. Moreover, in order to give ratings to their neighbors, nodes can use a watchdog mechanism to observe the behaviors of their neighbors thanks to the broadcast nature of messages. Note also that the utilization of second-hand reputation can influence the efficiency of these mechanisms: if the second-hand reputation is not allowed to be used, a sender node should be sure that all nodes on the route it will use are benign nodes (in case of source routing). However, due to the lack of organization in MANET, it could be a condition difficult to meet. Otherwise, if second-hand information is used, the security of their exchanges, the way to avoid lying attacks, are issues remained to be resolved. Furthermore, the way to ensure the authentication in the monitoring (supervision) is not studied. Thus, we believe that both watchdog and the exchanges of reputations need to be secured and be more efficiently integrated into the underline routing protocol.
We also noticed that MANETs need different routing protocols for different scenarios and applications. This diversity requires that both the proactive routing and the reactive routing be secured.

In the following, we clarify our research considerations that will be further developed in the rest of this thesis.
First, we intend to address the authentication problem in the watchdog mechanism. This problem is mostly ignored by the mechanisms using watchdog which assume that the authentication is ensured and the user identities are unspoofable and unforgeable. To relieve these hypotheses, and to reduce the storage requirement of watchdog, we suggest a mechanism called SWAN which is able to provide a lightweight broadcast authentication and an efficient storage scheme to watchdog without damaging its capacity of misbehavior detection.
Then, we consider the possibility of establishing a reputation system integrated into routing. We believe that observing what nodes really do is the best way to establish and maintain trust relationships between nodes, as well as to detect compromised nodes. Trust relationships can then be used in many aspects of MANETs, such as key management and secure routing.
Note that many reputation systems presented in section 3.5 assume that the routing is already secured by a certain secure routing protocol without specifying the nature of the protocol. Since SRP is a very lightweight secure routing protocol based on the source routing, it could be an appropriate underline routing protocol for supervision systems. We integrate a reputation system into it for proposing the TRP protocol which is also able to avoid the blackmail attacks.
The last problem that we consider in this thesis is the security of the OLSR proactive routing protocol. Note that the current proactive secure routing protocols are either not secured against the compromised nodes, or at the price of significant traffic and computational overhead. We thus propose a lightweight mechanism, which can prevent the link spoofing attacks from compromised nodes and ensure the integrity of the topology of the whole network, while still minimizing the security cost.

Chapter 4

SWAN: A Secured Watchdog for Ad hoc Networks

"A small miss in the beginning will cause a great error at the end."

– Unknown

4.1 Introduction

The watchdog mechanism [MGLB00] is widely used in the existing security mechanisms that are presented in the previous chapter. It is especially useful for the selfishness prevention in MANETs, since cryptography alone cannot achieve to prevent selfish behaviors. Thanks to watchdog, a node can detect its misbehaving neighbor nodes, and then supply such information to a reputation system which permits to isolate and/or punish misbehaving nodes. CORE [Mic04], CONFIDANT [BB02b], etc..., (c.f. section 3.5.3) are such examples which use watchdog.
In detail, the watchdog mechanism uses the promiscuous mode to help nodes to receive all the messages passed by their neighborhood. The received messages can then be analyzed in order to check whether or not there are misbehaviors out of line with the (routing) protocol in use. Malicious attacks and selfish behaviors committed by the neighbors can thus be detected.
In comparison with the other security mechanisms, the watchdog mechanism has the advantage of having neither additional traffic nor significant computational overhead. However, when using watchdog, nodes need to temporarily store many messages to perform bad behavior detection, which is a problem especially when the network is dense or when there is a lot of data traffic.
Moreover, watchdog needs to be secured against spoofing attacks since the latter can cause mistakes in reputation systems. For example, if an attacker X spoofs the identity of another node B when attacking, the reputation of B will decrease due to X. Most security mechanisms using watchdog introduced in chapter 3 assume that node identities are unspoofable but they do not mention any means to guarantee it. Some other mechanisms suggest authenticating every node during the route

discovery phase but leave the data forwarding phase unauthenticated, which will allow spoofing attack to take place during the data forwarding phase.
Finally, since there could be a large quantity of packets in the data forwarding phase, the authentication must be lightweight in order to support a large number of cryptographic operations.

In this chapter, we present an efficient mechanism to secure the watchdog. We refer to it as Secured Watchdog for Ad hoc Networks (SWAN). With SWAN, we provide the two following improvements to watchdog:

- First of all, to avoid spoofing attacks that may badly affect reputations, we combine SUCV [MC02] (c.f. section 3.3.2.4) and Timed Efficient Stream Loss-tolerant Authentication (TESLA) [PCSJ01, PCJS00] to provide a lightweight Broadcast Message Authentication (BMA) mechanism to watchdog.

  We give a brief introduction to SUCV and TESLA as follows:

  - A SUCV address naturally ties a Private Key (PK) and a node IDentifier (ID) together. Thus, no certificate server is required to establish a PKI. Furthermore, thanks to the use of IPv6 [DH98], it is proved that a SUCV address is statistically unique, thus being unspoofable.

    To realize the authentication and key management in SWAN, we use Hash chains to replace private keys in SUCV. As a result, the key pre-distribution and central key server are not required by SWAN, and each network identity becomes unspoofable.

  - Most secure proactive routing protocols [HJP02, HPT99, Che97] use either Hash chain or TESLA (c.f. section 3.4.2.1) to authenticate their routing messages and/or some control fields in their routing messages. Since the authentication of numerous messages can be done collectively, both Hash chain and TESLA provide a good way to handle a large number of messages in a lightweight manner.

    In addition, there is another approach called μTESLA [PSW+01] which principle is the same as TESLA but it assumes the existence of pre-established trust relationships and it uses symmetric keys to authenticate Hash chains. Thus, μTESLA is still more lightweight than Hash chain and TESLA and can be used by sensor networks (c.f. section 2.2.1).

    The above approaches inspired us to combine SUCV and TESLA to provide an authentication scheme to watchdog.

- Secondly, without loss of the observation capability, we propose an efficient storage scheme to reduce the storage overhead that is required by the classical watchdog. Instead of either storing a whole message (the case in [MGLB00]) or storing only the packet identity and a hash digest on the payload (the case in [Mic04]), SWAN stores the variable parts of a packet, a timestamp, and a hash digest on the fixed parts of the packet.

The chapter is organized as follows. We introduce in section 4.2 the notations used in this chapter, and then discuss the related work in section 4.3. In section 4.4 we give a full specification of SWAN, and then present some discussions in section 4.5. Finally, we conclude the chapter with section 4.6.
In chapter 5, we will show how SWAN can be applied to a secure routing protocol which uses watchdog.

4.2 Notations

In the following, we introduce the notations that are used in this chapter in their appearing order.

Notation	Meaning
X	a misbehaving node
A, B, C, E	nodes
T_Max	upper bound of the lifetime of the network
Δt	the duration of a time interval
$T0$	the network starting time
$n_1, n_2, ..., n_N$	the nodes in an N-node network
$\mathfrak{L}$	the length of a Hash chain
$[x/y]$	y integer divides x
s_i	a random seed chosen by node n_i
$h^j(a)$	a value a hashed j times without key
$hash - i(a)$	the i-bit hash output of a
τ	a time interval
τ_i	the ith time interval
M	a message
M_Fix	the fixed fields in the message M
M_Var	the variable fields in the message M
TTL	a Time To Live
HC	a hop count
$h_{key}(a)$	HMAC computed on a value a using the key key
I_i	the ith intermediate node on a route
$i\|j$	the catenation of i and j
IP_A	the IP address of node A
l	the length (in bits) of Hash values
N	number of nodes in the network
${}^xP_y$	number of permutations of x elements taken y at a time
${}^xC_y$	number of combinations of x elements taken y at a time
$h(a)$	the Hash value of a
c_j	the jth cycle in the network
$s_{i;j}$	a random seed chosen by node n_i for the cycle c_j
$IP_{A;j}$	the IP address of node A for the cycle c_j
$R_{i;j}$	the root of Hash tree of node n_i in the cycle c_j

4.3 Related work

In this section we mainly discuss some authentication mechanisms used by reputation-based security protocols which employ watchdog.

In CONFIDANT [BB02b] (c.f. section 3.5.3.2), a PKI similar to PGP is self-organized. Thus, asymmetric cryptography can ensure the authentication of routing control messages (RERR, RREQ and RREP) and ALARM messages. Data packets are implicitly supposed to be sent on routes that are discovered and chosen for this purpose. In other words, the nodes that forward the data packets are supposed to be the nodes discovered within the routing discovery phase. However, this hypothesis may not be always true, since in a mobile network a malicious node can pretend to be another node by spoofing the identity of the latter, and then attack the data traffic. As a result, attacks committed by the malicious node can decrease the reputation of the spoofed node. In CONFIDANT, no further mechanism is designed to address this issue.

In CORE [Mic04] (c.f. section 3.5.3.1), it is supposed that all identities are unspoofable and unforgeable. Later, the authors have proposed a key management and message authentication scheme called IDHC. This scheme relies on an offline Key Distribution Center (KDC) server to provide one ID-based master ticket to each node. Afterwards each node should generate a series of authentication tickets based on its master ticket. The tickets are then used in a way similar to the use of Hash chains in μTESLA: each ticket is used during one time interval and the authentication of packets is delayed to the next time interval. However, IDHC has a drawback in terms of computational overhead, since the generation of one authentication ticket is comparable to a RSA [Lab02] encryption, and the verification of a ticket is equivalent to a RSA signature verification.

As mentioned in section 4.1, usual methods for BMA that we found in the literature are Hash chain [HPT99], TESLA [ZXSJ03, Che97], μTESLA [PSW+01], Merkle Hash Tree [CBH03], etc... Among them, TESLA uses asymmetric keys to sign the first elements of Hash chains, and μTESLA uses symmetric keys to authenticate its Hash chains thanks to some pre-established trusts in sensor networks. Compared to them, the most important advantage of SWAN will be that it does not require a PKI or shared keys to authenticate the first elements of Hash chains. Furthermore, SWAN is carefully designed to be adaptive to reputation-based ad hoc secure routing protocols.

4.4 SWAN scheme

The main objective of SWAN is to provide a lightweight message broadcast authentication scheme to the watchdog mechanism against the address spoofing attacks. To achieve this goal, every node should firstly create an ID – a temporary address

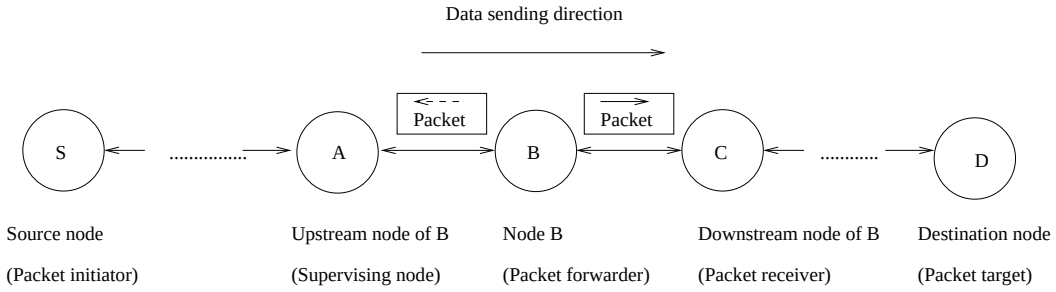


Figure 4.1: An example of supervision

based on a Hash chain, and then the TESLA authentication can be performed.
The assumptions of SWAN are described in section 4.4.1, and the scheme of SWAN is specified in section 4.4.2. The system requirements and a security analysis of SWAN are discussed respectively in section 4.4.3 and 4.4.4. Finally, we show the address renewal possibilities in section 4.4.5.

4.4.1 Assumptions

Neither additional module nor a-priori key distribution or key server is required by this scheme. In addition, nodes are not required to execute any asymmetric cryptographic operation. Nevertheless, nodes should be able to accomplish hash operations with a collision-resistant hash algorithm.
In addition, we suppose that the promiscuous mode is available to all nodes in the network, and IPv6 is in use.
Figure 4.1 shows an example of supervision in which node A supervises node B that is forwarding a packet to node C. In order to increase the accuracy of the supervision as possible as we can, it is further assumed that all nodes in the network have the same transmission range and that they all use an omni-directional antenna. Thus, all links would be bidirectional, and a forwarding node will always be heard by its upstream node except if there is a rupture of the link between the two nodes due to mobility. CSMA/CA RTS-CTS [Soc05] can be used as the media access control protocol to reduce the "hidden node problem" [MGLB00].
Regarding the storage, nodes are supposed to be able to store at least one Hash chain and to have a watchdog buffer in which unverified messages can be temporarily stored until the disclosure of their verification keys (c.f. section 4.4.3.2 for the storage requirement analysis of SWAN).
With the source routing algorithm a node can easily predict the entire data packets that should be forwarded by its downstream node, thus the source routing can facilitate the monitoring, and we consider it as our underline routing algorithm. This choice is the same to that of many other reputation-based solutions such as CONFIDANT and CORE. As a result, adopting SWAN to them is supposed to be easy.
For the routing algorithms other than the source routing, the adoption of watchdog is less obvious. Since a node is not sure of the next node of its downstream node,

it cannot easily judge if a packet is misdirected. Nevertheless, SWAN can also be used to support other types of routing protocols such as AODV and OLSR, as long as a watchdog is employed.

The following parameters are initialized and published in the network before SWAN is applied:

Lifetime of network T_Max: We assume that ad hoc networks are temporary and local area networks, and it is possible to estimate the upper bound of network lifetime T_Max[1].

Time interval duration Δt: The network lifetime is split into time intervals of uniform duration Δt.

Network start time $T0$: $T0$ serves as a reference to which all nodes synchronize their schedule of changing and disclosing keys.

All the above parameters are not secrets and can easily be published in the network. Besides, if the network is isolated, its IPv6 prefix can be chosen by itself[2]. Thus, these parameters can be integrated into the IPv6 prefix (according to the definition of IPv6, a unicast IPv6 address should start with "001"):

$$\text{IPv6 prefix} =< (3\ bit)001, (32\ bit)T0, (16\ bit)T\_Max, (13\ bit)\Delta t >$$

Finally, we assume a loose synchronization which guarantees an upper bound on the maximum synchronization error. In practice, a MANET synchronization mechanism can be the *Time Synchronization Function* (TSF) [Soc05] or the solution proposed in [RK04] which better ensures the multi-hop synchronization for ad hoc networks. The basic idea of these mechanisms is that each node periodically broadcasts a beacon frame which announces its timer to its neighborhood. Then, upon receiving the beacons, each node adjusts its local timer to the fastest-running timer. The synchronization can also be guaranteed by the GPS system, which provides an accurate time precision down to nanosecond.

4.4.2 SWAN specification

We show the five steps of SWAN in figure 4.2, and we detail these steps in the following subsections.

[1]Even though large and long term ad hoc networks are also under study, according to the definition, MANETs should be temporary and local area networks.

[2]Generally routing is not used for communication inside a traditional network. It is useful only for traffic between two or more networks. Thus, to facilitate routing and to distinguish internal traffic from external one, all the nodes in a same network should use a same prefix. However, since every node in ad hoc networks is also a router and routing is used for internal communication, we may think that a unique prefix is not necessary for MANETs if it does not need to communicate with other networks.

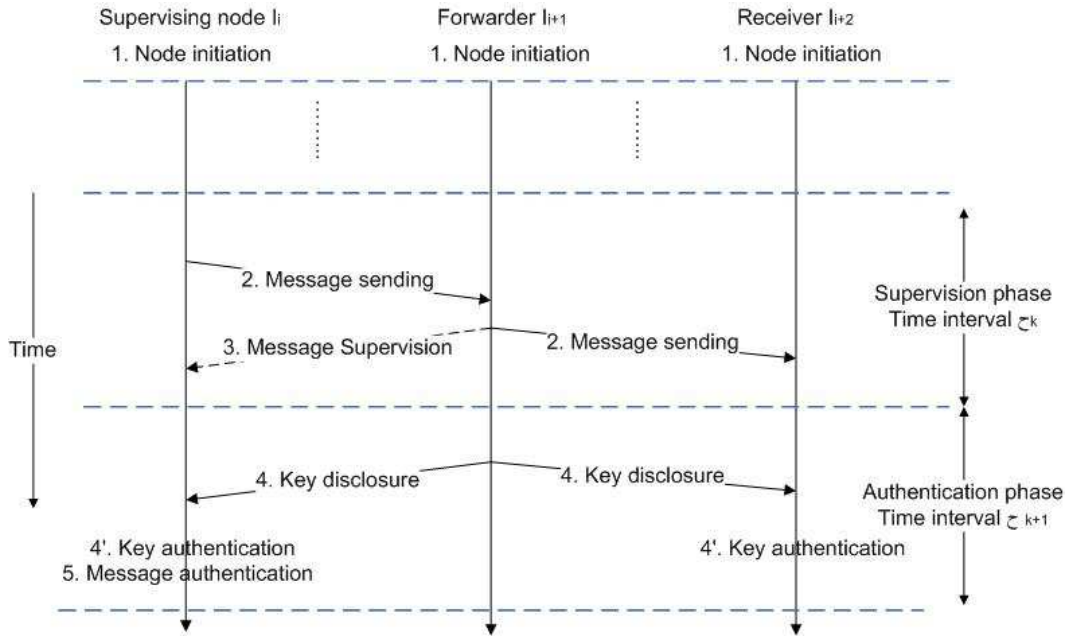


Figure 4.2: Steps of SWAN

4.4.2.1 Node initiation

The initiation of each node n_i should be done according to the following steps:

- Calculation of the length $\mathfrak{L}$ of Hash chains as $[T\_Max/\Delta t]+1$ ($[x/y]$ denotes y integer divides x).
- Generation of a Hash chain of $\mathfrak{L}$ elements based on a random seed s_i: $h(s_i)$, $h^2(s_i)$, ..., $h^{\mathfrak{L}}(s_i)$.
- Setting of the temporary address of the node:

$$IP_i =< \text{IPv6 prefix}, \text{hash-64}(h^{\mathfrak{L}}(s_i)) >$$

  where the last hash output length can also be reduced to 63 or 62 or even less, according to the length of reserved bits in the IPv6 header.

Thus, the network time is divided into $\mathfrak{L}$ intervals τ_1, ..., $\tau_{\mathfrak{L}}$ (only the last interval could be shorter than Δt). Any Hash chain element $h^{\mathfrak{L}-k+1}(s_i)$ $(1 \leq k \leq \mathfrak{L})$ will be used for the authentication of all messages sent or forwarded by node n_i during the time interval τ_k.

4.4.2.2 Message sending

Let M_Fix be the fixed IP header fields (address of the initiator, address of the target, the packet identifier, etc.) and payload (if any) of a message M, and let

M_Var be the mutable fields of M (TTL, HC, etc.). In time interval τ_k, M sent by node n_i will be in the following form:

$$< M, h_{h^{\mathfrak{L}-k+1}(s_i)}(h(M\_Fix)|M\_Var) >$$

Without loss of generality, M can be either a control packet or a data packet.

4.4.2.3 Message supervision

Suppose that in time interval τ_k, node I_i sends a packet M to node I_{i+1}, and then I_{i+1} should resend/forward M to node I_{i+2}. Then the first message sent by I_i will be in the following form:

$$< M, h_{h^{\mathfrak{L}-k+1}(s_i)}(h(M\_Fix)|M\_Var) >$$

I_i keeps the packet identity of M, $h(M\_Fix)$, and all the mutable fields of M in its watchdog buffer.
Upon receiving the first message, I_{i+1} performs necessary modifications (if any), and then the modified packet M' is sent to node I_{i+2} using the form:

$$< M', h_{h^{\mathfrak{L}-j+1}(s_{i+1})}(h(M'\_Fix)|M'\_Var) >$$

where j is the current time interval index according to node $i+1$.
The packet M' sent by I_{i+1} will be observed by I_i. I_i identifies the packet with its identity and finds the corresponding M in its watchdog buffer. I_i then checks M' mutable fields to see whether all the modifications realized by I_{i+1} respect the routing protocol in use (whether a TTL is decreased by one, etc...). Finally, I_i checks whether $h(M\_Fix) = h(M'\_Fix)$ for the integrity of the fixed fields of M. If all the above verifications are successful, we consider that the supervision is successful. I_i can then increase its reputation on node I_{i+1}. Otherwise, a suspicious activity of node I_{i+1} is discovered and we have the following choices:

- I_i can wait for the authentication phase to try to eventually identify I_{i+1} as a malicious node. This measure can prevent false negatives if nodes spoof when attacking.
- I_i can directly decrease the reputation of node I_{i+1} without further verification. This measure can prevent false positives even if nodes spoof when attacking.
- I_i can directly delete the packet from its watchdog buffer and leave the reputation for I_{i+1} unchanged. This measure does not decrease reputations but only increases them when there are successful supervisions.

With the first choice, I_i should further store $h_{h^{L-j+1}(s_{i+1})}(h(M'\_Fix)|M'\_Var)$ for the future authentication.
Above we described a supervision example using the mode SUpervision on ROute (SURO), in which only nodes involved in traffics observe what happens on routes. Actually SWAN can also support the mode SUpervision in NEighborhood (SUNE), in which any node that is neighbor of both I_i and I_{i+1} can perform the supervision.

4.4.2.4 Key disclosure and authentication

At each time interval τ_{j+1}, any node i checks if it has sent any message during the previous time interval τ_j. If so, it discloses its key $h^{\mathfrak{L}-j+1}(s_i)$ by broadcasting a Key Disclosure message (KD) to its one-hop neighbors after a maximum synchronization error:

$$< h^{\mathfrak{L}-j+1}(s_i) >$$

Note that if there is any periodic neighbor discovery process[3], keys can be disclosed within neighbor discovery packets by setting Δt equals to the neighbor discovery interval. This can reduce the control overhead.
Upon receiving a KD message, the receiver checks at first whether the corresponding key is already validated. If it is the case, it rejects the message. Otherwise, it verifies whether

$$IP_i =< \text{IPv6 prefix}, \text{hash-64}(h^{j-1}(h^{\mathfrak{L}-j+1}(s_i))) >$$

If the check fails, the key is rejected, and the node will still verify the other KD messages declaring the same key. Otherwise, the key is authenticated and stored, and the previous key discovered by node I_i is replaced by the new key.

4.4.2.5 Message authentication

Once a key $h^{\mathfrak{L}-j+1}(s_{i+1})$ is authenticated by node I_i, I_i checks in its watchdog buffer whether there is any message M sent by node I_{i+1} unauthenticated. If it is the case, the validity of $h_{h^{\mathfrak{L}-j+1}(s_{i+1})}(h(M\_Fix)|M\_Var)$ is checked.
If both the supervision and the authentication are successful, a good behavior is registered in favor of node I_{i+1}. Otherwise, if only the authentication succeeds, a bad behavior will be attributed to node I_{i+1}. Thus, to have a good reputation, nodes have to behave well in both routing and key disclosing.
If the authentication fails, it is possible that M is not sent by the node it claimed to be (in other words, there is a spoofing attack). Unfortunately, we are not able to identify the attacker. Therefore, to avoid false negatives, the result of the supervision may not be taken into account by the reputation system.

Note that SWAN mainly permits to detect malicious behaviors, but a selfish forwarding node[4] cannot be thus detected this way, since no message will be forwarded by the node. To resolve the problem, we suggest combining a solid neighbor lookup protocol with SWAN. Thus a neighbor node which does not regularly forward messages can be considered a selfish forwarding node.
We show an example of the whole SWAN process in figure 4.3.

[3] A periodic neighbor discovery process exists in most of the proactive and hybrid ad hoc routing protocols and in some secure ad hoc routing protocols.

[4] Refer to section 3.5.2. Selfish routing nodes do not participate in the routing discovery phase nor in the data forwarding phase, while selfish forwarding nodes only refuse to forward data packets.

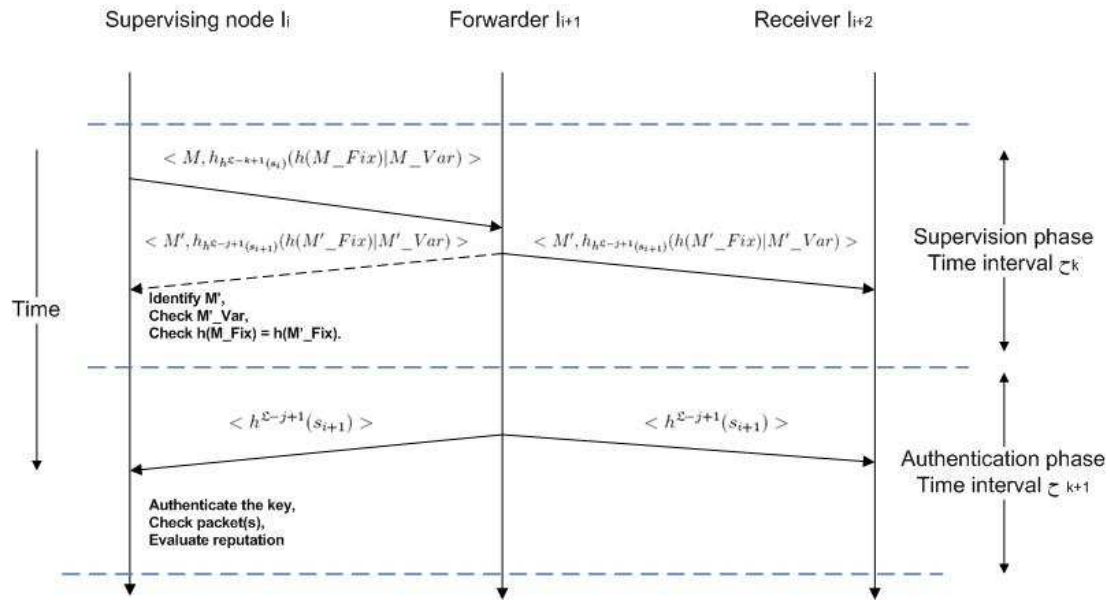


Figure 4.3: An example of SWAN

4.4.3 System requirements

In this subsection, SWAN is analyzed in terms of computational and storage requirements.

4.4.3.1 Computational requirements

Thanks to the temporary nature of MANETs, in SWAN no asymmetric cryptography is required even in the node initiation phase. The only cryptographic operation used in SWAN, the hashing, is known as lightweight[5].
Suppose that a 128-bit hashing is used to generate the Hash chains, and the length of seeds $s_1, ..., s_N$ is also 128 bits, we can then refer to table 4.1 to know the number of hash operations and the total hashed bits of each SWAN operation (note that in a real environment, the total hashed bits depends on the implementation of the hash algorithm).

Operation	Number of hash operations	Total hashed bits
Node initiation	$\mathfrak{L}+2$	$64+128*(\mathfrak{L}+1)$
Packet sending	2	$128+\text{packet\_length}$
Packet supervision	1	$\leq \text{packet\_length}$
Key authentication	j	$128*j$
Packet authentication	1	$128+\text{packet\_var\_length}$

Table 4.1: Hashing required by SWAN

[5]For example, the velocity of 160-bit SHA-1 one-way hash function is 75MB/s on a 33MHz 486SX.

Furthermore, by using 128 bits to store a before revealed key $h^{\mathfrak{L}-j+m}(s_i)(2 \leq m \leq j)$, we can decrease the number of hash operations required by the authentication of the key $h^{\mathfrak{L}-j+1}(s_i)$ to $m-1$, by checking whether $h^{\mathfrak{L}-j+m}(s_i) = h^{m-1}(h^{\mathfrak{L}-j+1}(s_i))$.

4.4.3.2 Storage requirements

We can refer to table 4.2 to see the number of bits to be stored by SWAN for each SWAN operation (under the same hypothesis as in subsection 4.4.3.1):

Operation	Length of total information to be stored (bits)
Hash chain storing	$128 * ([T\_Max/\Delta t] + 1)$
Packet authentication and supervision	length(Timestamp $+ 2 * 128 +$ packet_identity)

Table 4.2: Memory required by SWAN

To further reduce the memory space required to store a $\mathfrak{L}$-element Hash chain from $O(\mathfrak{L})$ to $O(log(\mathfrak{L}))$, the technique proposed in [Jak02] can be used. This technique selectively stores a logarithmic number of Hash chain elements, and the locations of the stored elements are modified over time, in such a way that we can easily find any Hash value through several hash operations.

4.4.3.3 Overhead

In SWAN the only additional message with respect to the routing protocol is the key disclosure message. During the lifetime of a MANET, each node sends at most $[T\_Max/\Delta t]$ KD messages to its one-hop neighbors, and each KD message has less than 150 bits as length (the MAC and IP header not taken into account).
In addition, every traditional routing message will have an additional overhead of 128 bits.

4.4.4 Security analysis

In this subsection, we show that SWAN is able to achieve its authentication objectives: either false negative or false positive in reputation system can be prevented, and messages can easily be supervised hop-by-hop. Moreover, the integrity of packets is guaranteed and the replay attacks are limited. In the end of the subsection, we also discuss the problem of bogus address in SWAN.

4.4.4.1 Reputation system security

We can ensure that at least the first-hand reputations for benign nodes will be correct, because:

- We suppose that a benign node will always perform correct routing operations with its true identities (IP and MAC addresses) and will always disclose correct

authentication keys. Therefore its reputations will certainly be increased by the other benign nodes observing it.

- Since SWAN guarantees that there is no spoofing attack, attackers are not able to decrease the reputations of benign nodes by spoofing their identities.

We distinguish two cases of misbehaving nodes:

- If a malicious node uses its true identity and its true keys to attack, benign nodes can identify it and decrease the reputations on it. Therefore there would be neither false negative nor false positive in the reputation system. However, in order to keep at least its bootstrapping reputation, the malicious node may prefer to belong to the next category.

- If a malicious node does not use its true identities to commit attack, or uses some incorrect keys to generate its attacking messages, the observing nodes will not be able to identify it. Thus, the reputation on the malicious node might not be decreased. However, since the reputations of benign nodes do increase, the malicious node will still have its reputations inferior to those of the benign nodes.

  Furthermore, since such a malicious node will always have its reputations unchanged, we may consider that, if a node existed in the network for a long period always has its bootstrapping reputations, it is less trustworthy than a new coming node or a node having its reputation evaluated to a higher level. This measure can also be used to detect the selfish routing nodes.

  Another solution is to let each node periodically decrease all its reputations on other nodes. Thus, even though we cannot authenticate the misbehaving nodes, they cannot keep their bootstrapping reputations, and every node must proactively participate in the routing to obtain/maintain good reputations.

Considerations for the reputations of selfish nodes

It is obvious that selfish nodes cannot be directly detected by SWAN, since no (less) message will be routed by them. Therefore, they do not have many messages to be authenticated.
Nevertheless, we distinguish selfish routing nodes from selfish forwarding nodes:

- For the selfish forwarding nodes, even without authentication we will be able to decrease their reputations since they refuse to forward data packets.

- For the selfish routing nodes, they can in the best case keep their bootstrapping reputation values, as the second type of malicious nodes discussed above.

As a conclusion, we believe that a reputation system helped by SWAN can provide relatively right reputations between nodes.

4.4.4.2 Statistically unique address

Since ad hoc networks are usually local area networks with a limited number of nodes (for example, DSR requires that the MANET dimension is less than 16), we believe that the SWAN addresses are, like SUCV addresses, statistically unique if the hash algorithm in use is *strong collision-resistant*. Here, strong collision-resistant means that, with an l-bit hash output, we need on average $2^{\frac{l}{2}}$ inputs to encounter a hash output collision.
Suppose that there are N nodes in a MANET, and that the hash algorithm that we use is a perfect l-bit hash algorithm. Then, the address collision probability of SWAN will be (let $W = 2^l$, $W >> N$, $N > 1$):

$$\begin{aligned}\text{Prob(collision)} &= 1 - \text{Prob(no collision)} = 1 - \frac{{}^{N}P_N \times {}^{W}C_N}{W^N} \\ &= 1 - \frac{W!}{(W-N)!W^N} = 1 - \frac{W-1}{W}\frac{W-2}{W}...\frac{W-(N-1)}{W} \\ &= 1 - (1-\frac{1}{W})(1-\frac{2}{W})...(1-\frac{N-1}{W}) \\ &< 1 - (1-\frac{N-1}{W})^{N-1} \sim 1 - (1-\frac{(N-1)^2}{W}) \sim \frac{N^2}{W}\end{aligned}$$

The collision probability increases with N and decreases with W (l). And, since $W >> N$, the collision probability is low.

4.4.4.3 Unspoofable address and authentication

To successfully spoof an IP address, the following methods can be tried by attackers:

Dictionary attack It is also called "brute-force attack" where attackers construct a database (also called a *dictionary*) which contains all the possible pairs of $< seed, h^{\mathfrak{L}}(seed) >$. Therefore once an $h^{\mathfrak{L}}(s_i)$ is revealed, attackers can look up the corresponding s_i in the dictionary. However, this attack is difficult to realize since it is equivalent to break the one-way hashing.

Replay It is somewhat true that without accurate time information in packets, the replay attacks can exist in MANETs. However, we believe that the replay attacks in SWAN cannot greatly affect the reputations of nodes. Indeed, we first do not take the messages replayed in the same time interval of their first sending into consideration; secondly, the messages replayed in a later time interval will be considered obsolete.

Finding a future key based on some revealed keys Even though Hash values will be revealed one by one by their owners, these attacks can be prevented since hashing is a one-way operation. The corresponding hashing property is called *weak collision resistance*, which means that given x, it is difficult to find a y that satisfies $h(y) = h(x)$. In other words, attackers are not able to find any unrevealed key from the revealed ones. This attack is also equivalent to break the one-way hash.

4.4.4.4 Integrity

Since both the mutable and the fixed fields of all the packets are protected by HMAC, the integrity is ensured in SWAN.

4.4.4.5 About bogus address

A malicious node can create a lot of bogus addresses in addition to its legitimate address. These bogus addresses will permit the node to bypass the reputation system by constantly appearing as a new node. Unfortunately, we can hardly prevent the existence of bogus addresses in absence of an online or off-line server.
Nevertheless, to complicate the generation of bogus addresses, we can use the binding of IP and MAC addresses as the identity of nodes. Since IP and MAC addresses are both unique and public, a node cannot solely modify its IP address without changing its MAC address at the same time. Furthermore, to obtain a new identity, an attacker should redo $\mathfrak{L}+1$ hash operations, which will also greatly complicate its task. Finally, since each bogus address is only temporarily used, a node using bogus addresses can hardly get a high reputation, thus having difficulty in becoming trustworthy.
We can also use the countermeasures against Sybil attacks presented in section 3.4.4.4 to detect bogus addresses in SWAN.

4.4.5 Address renewal

Although we suppose that we can estimate the maximum network lifetime in a most pessimistic way, a MANET can sometimes exist longer than expected. Otherwise, nodes could be too weak to support long Hash chains, or the lifetime of the network is too long to be supported by one Hash chain per node. In all these cases, addresses of nodes must be renewed but old reputations must not be lost. In other words, the old reputations should be related to the new addresses. In this subsection, we propose two mechanisms that seamlessly link a new Hash chain to the old one without introducing additional messages.

4.4.5.1 Approach using overlapping Hash chain

This approach consists of using two overlapping Hash chains, as shown in figure 4.4. During the node initiation phase, each node n_i picks two random seeds $s_{i;0}$, $s_{i;1}$ and generates two Hash chains: one chain of $\mathfrak{L}$ elements based on $s_{i;0}$, and the other of $2\mathfrak{L}$ elements based on $s_{i;1}$. n_i then sets its temporary address in *cycle* c_0 (in each cycle a node will use a different address, and the cycle c_0 is the first cycle): $IP_{i;0} =<$ IPv6 prefix, hash-64$(h^{\mathfrak{L}}(s_{i;0}))>$ and computes its address in cycle c_1 (the second cycle): $IP_{i;1} =<$ IPv6 prefix, hash-64$(h^{2\mathfrak{L}}(s_{i;1}))>$.
The message supervision is the same as the one described in section 4.4.2 except that I_i stores $h_{h^{2\mathfrak{L}-j+1}(s_{i+1;m+1})}(h_{h^{\mathfrak{L}-j+1}(s_{i+1;m})}(h(M'\_Fix)|M'\_Var))$ instead of $h_{h^{\mathfrak{L}-j+1}(s_{i+1})}(h(M'\_Fix)|M'\_Var)$.

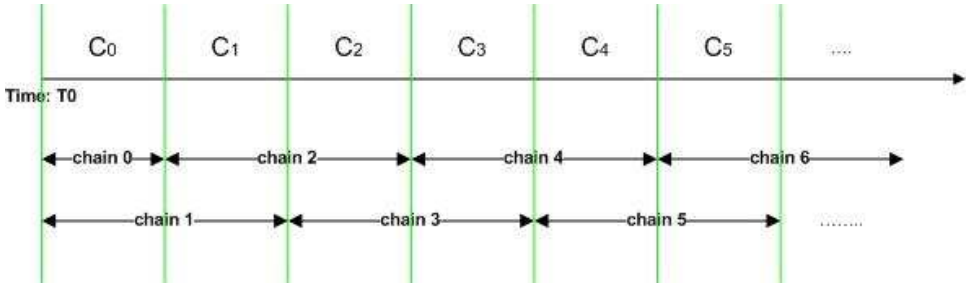


Figure 4.4: Address renew using overlapping Hash chains

The key disclosure and authentication in cycle c_m at time interval τ_{j+1} is as follows: node n_i publishes both $h^{\mathfrak{L}-j+1}(s_{i;m})$ and $h^{2\mathfrak{L}-j+1}(s_{i;m+1})$ in its Key Disclosure message:

$$< IP_{i;m+1}, h^{\mathfrak{L}-j+1}(s_{i;m}), h^{2\mathfrak{L}-j+1}(s_{i;m+1}) >$$

For the authentication of the keys, it is checked that:

$$IP_{i;m} =< \text{IPv6 prefix}, \text{hash-64}(h^{\mathfrak{L}+j-1}(h^{\mathfrak{L}-j+1}(s_{i;m}))) >$$

and

$$IP_{i;m+1} =< \text{IPv6 prefix}, \text{hash-64}(h^{j-1}(h^{2\mathfrak{L}-j+1}(s_{i;m+1}))) >$$

To authenticate messages, two HMAC operations have to be performed on $< h(M'\_Fix), M'\_Var >$ using successively the disclosed key pair $h^{\mathfrak{L}-j+1}(s_{i;m})$ and $h^{2\mathfrak{L}-j+1}(s_{i;m+1})$.
In the example quoted in section 4.4.2, the message authentication is successful if the corresponding computation result is equal to the stored $h_{h^{2\mathfrak{L}-j+1}(s_{i+1;m+1})}(h_{h^{\mathfrak{L}-j+1}(s_{i+1;m})}(h(M'\_Fix)|M'\_Var))$,
At time interval $\tau_{\mathfrak{L}}$ of cycle c_m, node n_i performs the following operations to renew its Hash chain: it picks a new random seed $s_{i;m+2}$, generates a Hash chain of $2\mathfrak{L}$ elements based on $s_{i;m+2}$, and then sets its temporary address in cycle c_{m+2} to:

$$IP_{i;m+2} =< \text{IPv6 prefix}, \text{hash-64}(h^{2\mathfrak{L}}(s_{i;m+2})) >$$

This approach can seamlessly link Hash chains together, and there is no additional overhead on payload. However, here each node has to store two Hash chains of $2 \times \mathfrak{L}$ elements (except in the first cycle in which each node stores one chain of $\mathfrak{L}$ elements and one chain of $2\mathfrak{L}$ elements) instead of storing one Hash chain of $\mathfrak{L}$ elements as in the original SWAN. In addition, one more HMAC should be computed when sending or authenticating a message, and KD messages also have a longer length.

4.4.5.2 Approach using Hash tree

In this approach, a Hash tree is established as shown in figure 4.5. The leaves of the Hash tree are IP addresses used in different cycles.
During the node initiation phase, each node n_i picks two random seeds $s_{i;0}$, $s_{i;1}$ and generates two Hash chains of $\mathfrak{L}$ elements. Then, it sets its temporary address

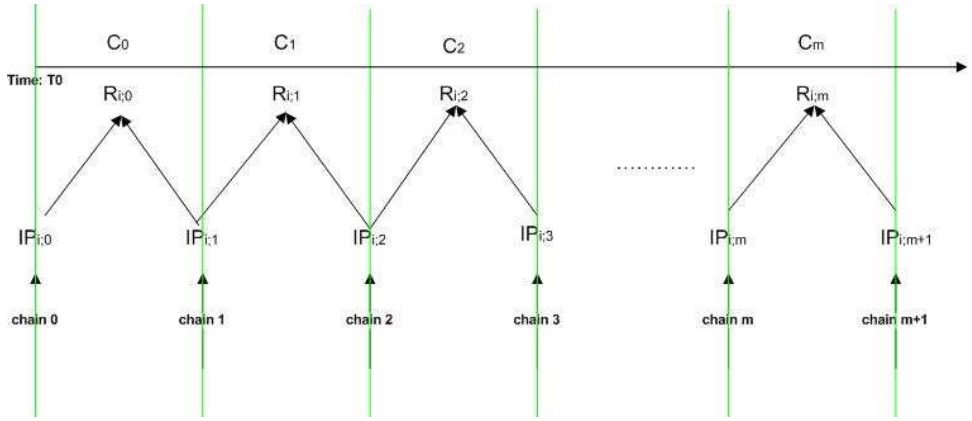


Figure 4.5: Address renew using Hash tree

in cycle c_0: $IP_{i;0} =<$ IPv6 prefix, hash-64$(h^{\mathfrak{L}}(s_{i;0}))>$ and its address in cycle c_1: $IP_{i;1} =<$ IPv6 prefix, hash-64$(h^{\mathfrak{L}}(s_{i;1}))>$. The root of the Hash tree in cycle c_0 is $R_{i;0} =$ hash-64$(IP_{i;0}|IP_{i;1})$.
In the kth interval of cycle c_m, the format of a packet M sent by the node n_i is:

$$< M, R_{i;m}, h_{h^{\mathfrak{L}-k+1}(s_i)}(h(M\_Fix)|M\_Var|R_{i;m}) >$$

where $R_{i;m}$ is the root of the Hash tree in cycle c_m.
The message supervision process is the same as the one described in section 4.4.2, since $R_{i;m}$ is regarded as a fixed field. The message authentication in cycle c_0 is also the same as the one described in section 4.4.2.
In cycle c_m $(m \geq 1)$, a Key Disclosure message will be

$$< IP_{i;m-1}, R_{i;m-1}, h^{\mathfrak{L}-j+1}(s_{i;m}) >$$

To authenticate the key, three verifications are necessary:

1. $IP_{i;m} =<$ IPv6 prefix, hash-64$(h^{j-1}(h^{\mathfrak{L}-j+1}(s_{i;m})))>$.
2. $R_{i;m-1}$ is the same as the root published in cycle c_{m-1}.
3. $R_{i;m-1} =$ hash-64$(IP_{i;m-1}|IP_{i;m})$.

In time interval $\tau_{\mathfrak{L}}$ of cycle c_m, node n_i picks a new random seed $s_{i;m+2}$. It renews its Hash chain by generating a Hash chain of $\mathfrak{L}$ elements based on $s_{i;m+2}$, and then sets its temporary address in cycle c_{m+2} to $IP_{i;m+2} =<$ IPv6 prefix, hash-64$(h^{\mathfrak{L}}(s_{i;m+2}))>$. The new root of the Hash tree in cycle c_{m+1} will be:

$$R_{i;m+1} = \text{hash-64}(IP_{i;m+1}|IP_{i;m+2})$$

Compared with the approach using overlapping Hash chains presented in section 4.4.5.1, this approach achieves its objective by adding more message overhead. However, it introduces less computational overhead and has less storage requirement.

4.5 Discussion

Choice of hash algorithm

The longer is the hash output length, the heavier is SWAN but the better is its security. We estimate that a 64-bit or longer hash algorithm is sufficient to reach the security requirements of SWAN. However, recent progress in the cryptanalysis on MD5 and SHA-1 [WY05, Len05, Ste06] leads us to expect stronger hash algorithms. In the simulations about SWAN presented in the next chapter, we use 128-bit hash output.

Synchronization

The synchronization is a common requirement of many secure ad hoc routing protocols such as ARAN [SDL+02] and SEAD [HJP02]. SWAN and the HPLS protocol that we will present in chapter 6 also require the synchronization.
A good clock synchronization mechanism for MANETs should be distributed and does not depend on any specialized hardware. Moreover, it is worth mentioning that the synchronization mechanism itself should be secured in order to provide secured "real" time information to nodes.

New coming node and leaving node

In SWAN, leaving nodes do not take away any secret of network but only their personal secrets, so they can leave without influencing the security of the network. Furthermore, a node made off can return to the network with a resynchronization which will decide the number of time intervals (keys) to be skipped.
A new coming node should synchronize itself to the network by adopting the IPv6 prefix. It can use the value of $T0$, Δt and the current time t to compute the index of the current time interval, and then use the value of Δt and T_Max to compute its Hash chain and its identity.

Network dimension

In order to have a weak address collision possibility, we suppose that SWAN is applied to the MANETs that have a limited dimension. Unfortunately, SWAN will not be adequate for very large MANETs such as the networks described in [WZ02].

Duplicate address

We mentioned in section 4.4.4.2 that SWAN addresses are statistically unique. But, if ever we need to be certain of their uniqueness, the Neighbor Discovery Protocol (NDP) for IPv6 [NNS98] can be used to resolve the duplicate address problem.
Using NDP, a new node chooses an IPv6 address when joining the network. Then, it broadcasts its choice to the whole network within a *Neighbor Solicitation* message.

If there is already another node which is using the address, it will send back a *Neighbor Advertisement* message to the new node. As a consequence, the new node must make another choice. The process can be repeated several times until the new node can find an unused address.

Immediate authentication

If we need immediate authentication of routing control messages, the ARIADNE protocol [HPJ02] (c.f. section 3.4.2.1) can be used. When sending a RREQ, the sender can estimate the arrival time of the request to the destination node, and the intermediate nodes will then use the Hash values corresponding to that time, in order to compute their HMAC outputs. As a result, when the RREP message is being returned to the source node, the intermediate nodes can be authenticated with their disclosed keys.

Influence of mobility

In case of strong mobility, KD messages can be sent to more nodes (such as 2 or 3 hop neighbors) in order to increase the authentication rate, which can also increase the reputation evaluation velocity.

Participation to another network

An address is valid only for the current network. To participate to another network, nodes should be re-initiated.

Address renewal

We insist on the idea that Hash chain renewal should rarely occur in SWAN, and that it is better to use one Hash chain in the whole network lifetime than dividing the entire network lifetime into several cycles and use one Hash chain per cycle. This is because the address renewal introduces not only additional overhead and complexity, but also an important inconvenience due to the variation of the IP addresses. Even though each node can know the new IP addresses of its neighbors, it cannot easily know the addresses of remote nodes. In the following we provide a brief introduction to this problem.

We suppose that each cycle will be uniform and reasonably long, and that nodes do not change their identities within a cycle. Thus, the problem appears only when a cycle finishes and the next one begins.
When a proactive routing protocol is in use, there is periodic routing information exchanged within the whole network. Consequently, the new addresses can be exchanged within the routing messages before the end of each cycle.

When a reactive routing protocol is in use, a source node can modify its RREQ message by adding its address in the next cycle. Then, once being a source node, any node can inform all the other nodes about its next address.
If the next address of a destination node is unknown, a RREQ can still be sent to the old IP address (with a flag telling that it is an old address). Since the RREQ is broadcasted, the message will be received by the destination. Then, when a RREP is sent back, the new address can be joined.
Intermediate nodes usually have necessary knowledge about their upstream and downstream nodes since they are neighbors, and this would be sufficient for the supervision. Moreover, new address information can also be accumulated in a RREQ (like in DSR we accumulate node identities) when the end of a cycle approaches, and this could make all the new addresses on a route known by the whole route.
If there is an active data flow but the end of a cycle is reached, the source node can send an additional message along the route to collect the new addresses on the route.
Finally, NDP [NNS98] can also be modified to inform the new addresses to nodes. That is to say that each node can send out its new address in a *Neighbor Solicitation* message a little before the end of each cycle. If there is no new *Neighbor Solicitation* message during a timeout from the same node (means that the address is not duplicated), in the next cycle other nodes can use the new address to replace the older address of the node. Note that nodes should also adjust their message sending time in order to avoid collision.
After all, we see that the address renewal process is complicate to manage, and must as a consequence only be used when it is strictly necessary.

SWAN applicability

SWAN can operate with the source routing algorithm where every packet to be forwarded is perfectly predictable.
With the other routing algorithms, the receiver node of a packet can be decided on-the-fly by its upstream node. Thus a future packet is not entirely predictable from the viewpoint of the supervising node, and the watchdog is not able to check all the fields of the packet. However, the other fields of the packet except the receiver node can still be supervised. Since SWAN is a generic security mechanism independent of underline routing protocol, we believe that it can also be applied to the other routing algorithms.

4.6 Conclusion

In this chapter, we proposed a secure watchdog for ad hoc networks named SWAN. It combines SUCV and TESLA to develop a watchdog with a lightweight broadcast message authentication scheme. It can detect the spoofing attacks that may badly affect the reputation systems, and can reduce the storage overhead required by watchdog. It is also able to treat a large number of messages through a simple

cryptographic operation and be independent of any central server. Our analysis in this chapter and our simulations in the next chapter show that SWAN is both lightweight and robust.

In the next chapter, we will propose a secure routing protocol and apply SWAN to it, and we will also show some simulation results for SWAN.

Chapter 5

TRP: A Trust-based Routing Protocol for ad hoc networks

"Know your enemy and know yourself."

– Sun zi (about 535 B.C.)

5.1 Introduction

Generally, in ad hoc networks every node should participate in the routing. Therefore, each routing behavior of each node can have an impact on the routing security. Especially, a data flow can be threatened if the route established for a flow contains a misbehaving node.
In order to make every node be responsible for its own routing behaviors, a reputation system can be established. This system should reward the well-behaved nodes, and/or punish the misbehaved nodes. CORE [Mic04] (c.f. section 3.5.3.1) and CONFIDANT [BB02b] (c.f. section 3.5.3.2) are such examples. With them, each node can compute a reputation for each other node based on the routing behaviors of the latter. Thus, routes composed of nodes having better reputations can be chosen for the ad hoc routing, and/or data packets sent by misbehaving nodes may be dropped.
CORE and CONFIDANT are designed to guarantee the availability and the reliability of the ad hoc routing. However, CONFIDANT intend to exchange the reputations between nodes in a proactive way, which will introduce a lot of additional overhead to the network. Moreover, both protocols have not completely resolved the problem of the blackmail attacks caused by the use of second-hand reputations. In other words, liars can always introduce wrong reputations into the network, especially when they are numerous.
In this chapter, we present a DSR-based secure ad hoc routing protocol named Trust-based Routing Protocol (TRP). First, TRP uses an HMAC to protect routing control messages between each pair of initiator and target of data traffic. Second, it uses a watchdog to detect selfish behaviors and a few residual routing attacks untreated by the HMAC, in order to identify the misbehaving nodes with a reputation system.

Finally, based on the reputations, source nodes will be able to choose the most reliable routes to send their data packets.
In comparison with the other routing protocols using a reputation system, the main particularity of TRP is its reactive reputation exchange. That is, the exchange of reputation information is performed only when a route is needed to be used in the routing. Moreover, there is no additional packet used for the exchange of reputations. Instead, the exchanges are integrated into the routing control messages of DSR, and this integration can be made naturally since DSR is itself a reactive routing protocol. Finally, TRP is able to prevent blackmail attacks, even though the second-hand reputations can be taken into consideration.
In addition, TRP can also use SWAN (c.f. chapter 4) to secure its watchdog mechanism and improve its performance.
The remainder of the chapter is organized as follows. We introduce at first the notations used in this chapter in section 5.2. We provide an overview of TRP in section 5.3, and introduce the design objectives of the TRP protocol in section 5.4. In section 5.5 we discuss our reputation system. The TRP routing protocol is presented in section 5.6. In section 5.7, we show how SWAN can be applied to TRP. Section 5.8 is dedicated to the performance evaluation. Finally, our conclusion is presented in section 5.9.

5.2 Notations

In the following, we introduce the notations that are used in this chapter in their appearing order.

Notation	Meaning
$K_{A,B}$	a symmetric key shared by node A and node B
D	a destination node
I_i	the ith intermediate node on a route
S	a source node
$C_{A\to B}$	trust value that node A has on node B
N	the number of nodes in a network
$n_1, n_2, ..., n_N$	the nodes in an N-node network
$CD_{A\to B}(t)$	direct trust that node A has on node B at time t
α, β	parameters in the trust model of TRP
$p_{A\to B}(t)$	the experience level that node A has on node B at time t
$p^+_{A\to B}(t)$	number of positive experiences of node B that have been observed by node A until time t
$p^-_{A\to B}(t)$	number of negative experiences of node B that have been observed by node A until time t
$CDI_{A\to B}$	direct or indirect trust that node A has on node B
$CI_{A\to B}(t)$	indirect trust that node A has on node B at time t
d	a route length
A, B, C, E	nodes
r	a random integer

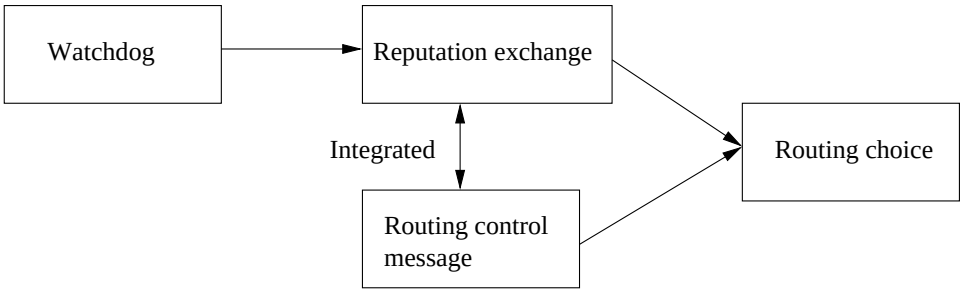


Figure 5.1: Basic idea of TRP

SN	a sequence number
$A \rightarrow * : M$	a node A broadcasts the message M to all nodes in the network
IP_A	the IP address of node A
$h_{key}(a)$	HMAC computed on a value a using the key key
$i\|j$	the catenation of i and j
M_Fix	the fixed fields in the message M
M_Var	the variable fields in the message M
τ_i	the ith time interval
$h^j(a)$	a value a hashed j times without key
UID	a unique identifier for each packet (in Network Simulator)
X_1, X_2, ...	misbehaving nodes

5.3 TRP overview

The Trust-based Routing Protocol is a DSR-based secure ad hoc routing protocol. It combines the knowledge of misbehaving nodes with topology information, to help source nodes to choose the most reliable routes for their data sending. We show the basic idea of TRP in figure 5.1.
In TRP, each node maintains a first-hand (direct) reputation for every other node that it has encountered. The computation of reputation is based on the directly observed behaviors of the other node. Then, during a route discovery process, intermediate nodes can inform the source node of their first-hand reputations on their neighbor nodes, by integrating them into the control messages (RREQ and RREP) of DSR.
A source node may receive a lot of RREPs (there is one route in each RREP) and a series of reputations for each RREP. Then, based on the received reputations and its own first-hand reputations, the source node can compute an overall reputation for each route. Only a route that has obtained an acceptable overall reputation can be trusted and be used to deliver data traffic.
The different phases of TRP are:

- Since a reputation system is a reactive system, and since we often suppose that there is no *a-priori* trust between nodes, misbehaving nodes will be able

to misbehave at the beginning of the network.

- Gradually, misbehaving nodes will have their reputations decreased, since their misbehaviors will be detected. They will be identified as attackers or failing nodes. On the contrary, benign nodes will have their reputations increased.

- With the evolution of the reputations, misbehaving nodes will have less and less possibilities to participate in the network routing, since they will be bypassed. Benign nodes, on the contrary, will be more frequently used in the routing.

TRP assumes that a pairwise key $K_{S,D}$ is shared within a security association between the initiator S and the target D of each RREQ message. Such a key can be established through several ways, such as DH [RLTN93], Internet Key Exchange (IKE) [HC98], etc.. Note that we do not assume a key shared by each pair of nodes. TRP uses a routing scheme similar to SRP (Secure Routing Protocol, c.f. section 3.4.2.2) [PH02]. In addition, it accumulates reputation information during the propagation of RREQq, and sends back these information protected by HMAC within RREPs to source nodes.

5.4 Design objectives

TRP aims to avoid misbehaving nodes in the two routing phases: the routing discovery phase (also called the topology discovery phase), and the data forwarding phase (the routing maintenance phase is included), by means of adding simple cryptographic operations and a totally distributed reputation system to the DSR protocol. Moreover, nodes which do not correctly forward data should be identified and excluded from routing.
TRP has the following security objectives:

- TRP should guarantee the authenticity of routing information. That is, under certain hypotheses, only correct route information can be received by source nodes. Indeed, the SRP protocol already guarantees this property in a great measure, and we will further reinforce it.

- According to the selfish models that are described in [Mic04] (c.f. section 3.5.2), TRP should be able to avoid the *selfish forwarding* behaviors, which are the most harmful selfish behaviors in MANETs.

  We believe that the *selfish routing* behaviors are hardly avoidable in a totally self-organized MANET using a reactive routing protocol, because in such a network a node can simply be silent to escape its routing duties, and normally no mechanism permits to detect such selfish nodes. To resolve this problem, each node can regularly reduce the reputations of the other nodes, which makes a *selfish routing* node not being able to keep good reputations by the other nodes. Besides, if the network is to some degree organized in such a way that each node inside the network is known, selfish routing nodes will be determinable (however we may still need a central server to determine them).

- TRP should permit to prevent a large number of active attacks in the ad hoc routing.

In TRP, a trade-off lies between the routing performance and the security, since we try to provide a maximum security while still maintaining an acceptable routing performance. We require that, without the presence of attackers, TRP should have a good routing performance; with the presence of attackers, TRP should be much better than an unsecured ad hoc routing protocol in terms of routing performance.

5.5 Reputation system

Due to the existence of compromised nodes that are able to reply correctly to cryptographic challenges thanks to their legitimate keys, cryptography alone cannot ensure the security of the mobile ad hoc networks.
Since the cryptography is not almighty, the notion of trust is introduced into MANETs. With this notion, nodes in MANET are called up for vigilance. In order to engage only trustworthy nodes into routing, in a totally distributed MANET each node should maintain a trust level to each other node.
With this framework, reputation systems can be applied to MANETs. As opposed to key/cryptography systems that are often established in advance, reputation systems are systems that will only be established *a-posteriori* except that there could be some pre-established reputation relationships. By default every node is attributed a same initial reputation value at the beginning of the network.
Then, as time progresses, it should be ensured that:

- Benign nodes and misbehaving nodes are correctly identified.
- A misbehaving node will have a lower reputation than a benign node.
- There should be less and less misbehaviors in the network, since low reputed nodes will be ruled out or be stimulated to behave well.

The original model of the reputation system used in TRP is a distributed trust model introduced in [YB94] and later developed in [BBK94] for its valuation part. Since it is not initially designed for ad hoc networks, we modified it for TRP. In the two following subsections, we introduce respectively the original model and our model.

5.5.1 Original reputation model overview

This model [YB94, BBK94] allows us to take various classes of trust relationships into account, in such a way that an entity can be trusted for some specified tasks (also called functions) but not for some others. For instance, a node trusted for nondisclosure of secrets may not be trusted for generating keys due to its weak computational capacity.

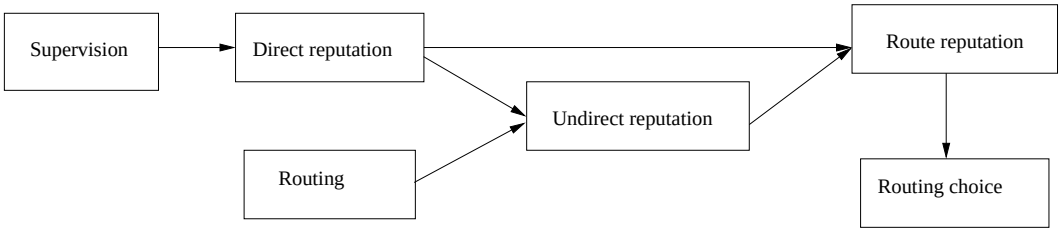


Figure 5.2: The relationships between the different types of reputations in TRP

The model also allows the valuation of trust relationships. That is to say that according to the numbers of positive and negative experiences that an entity has assigned to another entity regarding a given function, a value can be computed and used as the trust level that the former entity assigns to the latter entity.
Finally, in the model it is also possible to derive indirect trust relationships using trust transitivity.

5.5.2 Our reputation model

In this section, we adapt the above model to ad hoc networks. Three types of trust relationships are considered in our model:

- *Direct trust from a node to a neighbor node.* It is evaluated according to the positive and negative experiences that the former node observed when supervising the neighbor node.
- *Indirect trust from a node to a distant node.* It is derived from direct trust relationships using transitivity, and it will be computed according to the routing requirements.
- *Trust from a source node to a route.* It can be computed based on both direct and indirect trust values, and it will be used to measure the trust levels of routes.

Figure 5.2 shows the relationships between the different types of reputations/trusts in TRP.
In the following, we define the computation formulas for the three trust types. All of the trust values are taken in the intervals $\{-1\} \cup [0, 1[$.

5.5.2.1 Direct trust

By default, all direct trust values are initialized to 0. However, since the model is totally local and distributed (thus each node can have its own independent trust values), nodes are free to initiate their trust values to the values they desired, especially when pre-established trust relationships exist.
Before defining the formulas used for the evaluation of the direct trust value from a node n_i to a neighbor node n_j, we define at first the following notations:

- $p^{+}_{n_i \to n_j}(t)$ is the number of positive experiences (the number of good behaviors) that n_j behaved and n_i observed until time t.
- $p^{-}_{n_i \to n_j}(t)$ is the number of negative experiences (the number of bad behaviors) that n_j behaved and n_i observed until time t. Note that in ad hoc networks, we have to tolerate a small percent of negative experiences due to the unreliable nature of ad hoc wireless medium.
- $\alpha \in]0,1[$ is a parameter that can be configured by the nodes themselves. We will see from the formulas that, higher is α, less is the influence of positive experiences on direct trust values (in other words, more positive experiences are required to make a direct trust increase from 0 towards 1). Thus, generally α should be set relatively high to prevent the (misbehaving) nodes from easily gaining a high reputation.
- β ($\beta > 1$) is a parameter that modulates the importance of negative experiences in relation to positive experiences. We will see from the formulas that, greater is β, larger is the influence of negative experiences on the direct reputations. Thus, same to α, β should also be set relatively high to prevent the misbehaving nodes from obtaining a high reputation.

The formulas to compute the direct trust value from a node n_i to a neighbor node n_j is defined as follows:

$$CD_{n_i \to n_j}(t) = \begin{cases} -1 & \text{if } p_{n_i \to n_j}(t) < 0 \\ 1 - \alpha^{p_{n_i \to n_j}(t)} & \text{otherwise} \end{cases}$$

and

$$p_{n_i \to n_j}(t) = p^{+}_{n_i \to n_j}(t) - \beta \times p^{-}_{n_i \to n_j}(t)$$

The above formulas can guarantee that:

- If a node always behaves well, trust values for it can be increased towards 1. Thus, such a node will be considered trustworthy.
- If a node is failing or moderately malicious, trust values for it will be more or less stable (however, this depends on the values of α and β). Thus, such a node is much less trustworthy than a node highly reputed.
- If a node is malicious or quite failing, trust values for it will rapidly be decreased to -1. Such a node is considered not trustworthy and should be avoided by the ad hoc routing.

With direct trusts, a network can be considered as a directed graph. This graph is complete only if direct trusts can be established between each pair of nodes in each direction. However, usually not all nodes have chances to be neighbors. Therefore the graph has a strong possibility to be incomplete. Therefore, it is necessary to introduce another type of trust, the indirect trust, for the estimation of trusts between distant nodes.

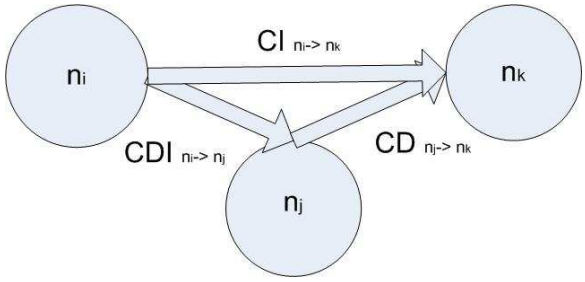


Figure 5.3: Indirect trust in TRP

5.5.2.2 Indirect trust

In figure 5.3 we show an example of the indirect trust computation in TRP. Suppose that there is a node n_k to which a distant node n_i has no direct trust relationship, but node n_i has a direct trust value or a reliable indirect trust value to another node n_j denoted $CDI_{n_i \to n_j}$, and node n_j has a direct trust value to node n_k denoted $CD_{n_j \to n_k}$. Then the indirect trust value from node n_i to node n_k denoted $CI_{n_i \to n_k}$ is defined as:

$$CI_{n_i \to n_k} = \begin{cases} -1 & \text{if } CDI_{n_i \to n_j} \text{ or } CD_{n_j \to n_k} = -1 \\ 1 - (1 - CD_{n_j \to n_k})^{CDI_{n_i \to n_j}} & \text{otherwise} \end{cases}$$

where

$$CDI_{n_i \to n_j} = \begin{cases} CD_{n_i \to n_j} & \text{if } p^+_{n_i \to n_j} + p^-_{n_i \to n_j} \geq 1 \text{ (if } n_i \text{ observed } n_j) \\ CI_{n_i \to n_j} & \text{otherwise} \end{cases}$$

If there are more than one possible node n_j, n_i will choose the n_j with the highest direct trust value $CD_{n_i \to n_j}$. Otherwise, if there is no n_j with a direct trust value $CD_{n_i \to n_j}$, n_i will choose the n_j with the highest $CI_{n_i \to n_j}$ to calculate $CI_{n_i \to n_k}$.
The above formulas can guarantee that:

- If one of the direct/indirect trust values in the recommendation chain (in the above example, the recommendation chain is n_i - n_j - n_k) indicates that n_j or n_k is a misbehaving node ($CD_{n_i \to n_j}$ or $CDI_{n_i \to n_j}$ equals to -1), then the derived indirect trust value $CI_{n_i \to n_k}$ will be equal to -1.

- If $CD_{n_j \to n_k}$ is based on an experience level $p_{n_j \to n_k}$, then $CI_{n_i \to n_k}$ is based on the experience level $p_{n_j \to n_k} \times CDI_{n_i \to n_j}$, since $CI_{n_i \to n_k} = 1 - (1 - (1 - \alpha^{p_{n_j \to n_k}}))^{CDI_{n_i \to n_j}} = 1 - \alpha^{p_{n_j \to n_k} \times CDI_{n_i \to n_j}}$.

The computation of indirect trust values is a necessary but intermediate step of trust computations in TRP. To choose reliable routes, a source node should be able to evaluate a trust on each of the available routes. Thus, we introduce the *route trust* in the next subsection.

5.5.2.3 Route trust

Suppose that any two communicating nodes (each pair of initiator and target) trust each other. Then, according to the principle that "a whole system is as strong as its weakest point", the trust level of a route is defined as the source's lowest trust level on all the intermediate nodes of the route.
For example, the trust value of a route S, I_1, ..., I_{d-1}, D is:

$$CR_{S \to S,I_1,...,I_{d-1},D} = min_{i=1}^{d-1}(CDI_{S \to I_i})$$

where for $1 \leq i \leq d-1$

$$CDI_{S \to I_i} = \begin{cases} CD_{S \to I_i} & \text{if } p^+_{S \to I_i} + p^-_{S \to I_i} \geq 1 \text{ (if } S \text{ observed } I_i) \\ CI_{S \to I_i} & \text{otherwise} \end{cases}$$

If a source node knows a lot of routes to a same destination, it can quantify the reliability of each route by using the above formulas, and then choose the route having the highest route trust value for its data sending.

5.5.3 Comparison with the original model

In the original model [BBK94], a node is judged malicious once it commits a misbehavior. However, due to the MANET nature, our model has to be fault-tolerate. Thus we set the parameter β to tolerate until $\frac{p^+}{\beta}$ negative experiences before judging that a node is malicious.
Moreover, we have added an additional value, -1, to keep a unique rating for all the untrustworthy nodes. Theoretically, all the misbehaving nodes should be marked -1, and they will have no possibility to be intermediate node on any route. Therefore they will not be able to threaten the data traffic initiated by the benign nodes.
Finally, we have introduced the notion of route trust into our model. It is used by the source nodes to quantify the secure level of each received route. Only the routes considered secure can be used in the ad hoc routing.

5.6 TRP routing

In this section, we discuss how our reputation system can be integrated into the DSR routing protocol to give birth to the TRP routing protocol.

5.6.1 Assumptions

The following assumptions are fairly common to the other protocols using a supervision system:

- We suppose that TRP is based on the basic functionalities of DSR, and most of the performance optimizations of DSR are removed. Nevertheless, we keep the supervision mechanism for our watchdog, which is initially used by DSR to facilitate the routing information collection for each node.

- For the "supervision on route" mode, each node should have at least a sufficient storage capacity for the supervision of a restricted part of the traffic forwarded by itself.

  For the "supervision in neighborhood" mode, each node should be able to save at least a restricted part of the traffic passing through its neighborhood.

  However, if SWAN is applied to TRP, the storage requirements can be reduced.

- Each node has a sufficient computational capacity for carrying out some cryptographic operations, in particular HMAC.

- The transmission ranges of nodes are identical. Moreover, we suppose that all radio links are bidirectional.

- Each node has a unique identifier (ID), and nodes can be identified during the supervision.

- Any two communicating nodes share a key. For example, between a sender node S and a destination node D, there could be a Security Association containing a shared secret key $K_{S,D}$. This SA can also be used to share other information between S and D, such as a random seed.

- There is at most one attacker on each route. Otherwise, even if two and more attackers can be on a same route, they are not neighbors and they cannot cooperate.

5.6.2 Routing discovery phase

In this section, we discuss the security mechanisms employed in the routing discovery phase of TRP. We show at first the cryptographic measures of TRP in subsection 5.6.2.1. Then, in subsection 5.6.2.2 we discuss how the reputation system presented in section 5.5 can be applied to TRP. Finally, the route management of TRP is presented in subsection 5.6.2.3.

5.6.2.1 Basic security mechanism

SRP largely inspired TRP for the security of the routing discovery phase. Figure 5.4 shows the header of SRP [PH02] integrated into DSR RREQ and RREP.
TRP also adds an additional header to the two DSR routing discovery messages. Like SRP, the additional header of TRP contains an HMAC code and two integers: a sequence number SN and a random integer r.
In SRP and in TRP, the basic routing discovery process is as follows:

- A sender S initiates a routing discovery phase by broadcasting a RREQ with the additional header, where the HMAC field is computed over the RREQ (except the *source route*) using the key $K_{S,D}$. As a consequence, all of the original fields in the RREQ are protected from alteration, and the message can be authenticated by the destination node which shares the key.

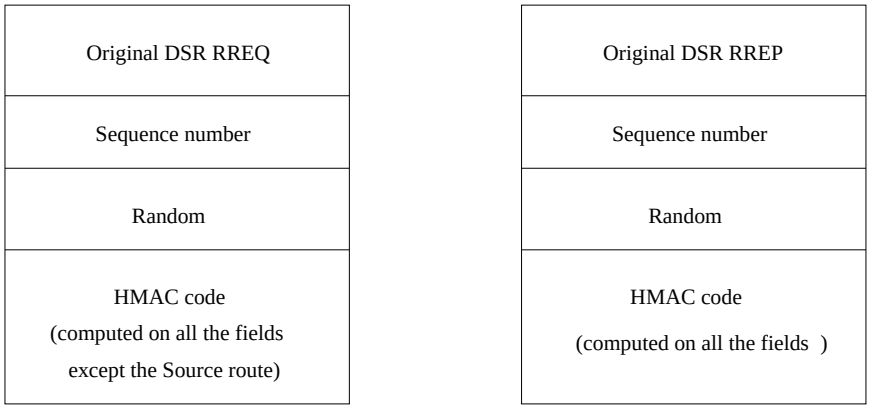


Figure 5.4: Header of SRP integrated into DSR RREQ and RREP

- During the broadcast of a RREQ, intermediate nodes add their identities to the request, and rebroadcast it until the latter reaches its destination. Note that intermediate nodes are not able to verify the authenticity nor the authentication of the message, and their added identities are not protected during the propagation of the RREQ.

- Upon receiving a RREQ, the receiver D verifies the HMAC code in it. If the verification is successful, D is sure that S wants to establish a communicating route with it. Then D sends back a RREP including the source route, the SN and the r received within the RREQ, and a new HMAC code computed over the entire RREP. Note that different to the case of RREQ, in a RREP the whole source route is protected against modification.

  Multiple RREPs will be sent to the initiator if multiple disjoint routes are found.

- Once a RREP reaches S, S verifies the HMAC code in it. If it is successful, the route included in the RREP is stored in the route cache of S, and can be used to send data.

In this process, the security is mainly ensured by the HMAC field. Meanwhile, SN is used to provide the information regarding the freshness of messages. In addition, it is also used to identify the messages especially for avoiding loops in the broadcast of RREQs.
The generation of r is based on a random seed that is usually shared within the SA between the two communicating nodes. r provides not only an additional possibility for the destination node to ensure the freshness of the message, but also an additional guarantee for the authenticity of the RREQ.
Additionly, when SN and r are sent back to the source node, they also contribute to the identification of the RREP towards the source node.

According to the detailed analysis shown in [PH02], the above mentioned mechanism is able to resist to a large number of attacks that the DSR protocol may encounter.

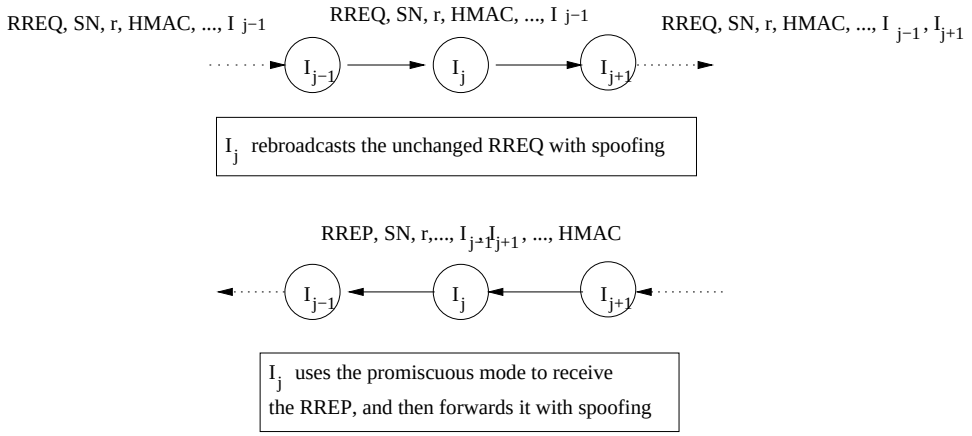


Figure 5.5: A vulnerability in SRP

Moreover, it is very lightweight since it only employs one additional HMAC operation per RREQ or RREP message.
However, in addition to the cooperating internal attacks that are originally announced as untreated by SRP, we found the following vulnerabilities in the mechanism:

1. Figure 5.5 shows an example of an attack against the mechanism. A malicious node, I_j in the figure, may refuse to add its identity to a RREQ but spoof the identity of I_{j-1} and rebroadcast the RREQ. Since by using the promiscuous mode I_j can receive the corresponding RREP and can spoof the identity of I_{j+1} when forwarding it towards the source node, the attack cannot be detected. Therefore, the source route in the RREP will seem shorter than its real length. The problem cannot be completely resolved by the NLP protocol neither, since NLP cannot totally prevent the spoofing attacks if the attacker changes both its MAC and IP addresses.

2. If an attacker is a neighbor of the destination node and if it rebroadcasts a RREQ many times by adding to it at each time a different spoofed IP address, multiple fake routes can be created. This attack can be realized since the destination node accepts routes from each different neighbor. However, if SRP is used with the SMT protocol, source nodes will choose only disjoint routes for their data sending. Therefore the problem can be avoided except for one case: there is only one intermediate node - the attacker - on the route.

3. A loop can be inserted into a RREQ because no verification is foreseen at this level. However, the loops can easily be detected and avoided during the propagation of the RREQs.

4. Selfish nodes are untreated. A selfish node can refuse to forward/rebroadcast control messages as well as data messages. However, SMT (c.f. section 3.4.2.3) can be used with SRP to guarantee the forwarding of data packets.

Except the third attack that is easily preventable, we try to counter the other misbehaviors by adding to it the reputation system described in section 5.5.

5.6.2.2 Reputation management in the routing discovery phase

In addition to the header introduced in section 5.6.2.1, TRP also uses another new header that is added to RREQs and RREPs. The new header can help to transport the trust information of the other nodes on routes to source nodes. This transportation will permit the initiators to compute a trust degree for each route they may use, and then choose the most reliable routes for their data sending.

Supervision system

According to the events observed through the watchdog, a node can dynamically calculate and update a direct trust value for each of its neighbors.
Each node maintains a *trust table* which memorizes the number of good behaviors p^+, the number of bad behaviors p^- and the direct trust value (also called "rating") for each of its observed neighbors. The calculation of the ratings follows the formulas defined in section 5.5.2.1.
To prevent the overloading of nodes, we choose a restricted supervision mode: the "supervision on route" mode. In other words, a node only supervises the messages past through itself.
In the routing discovery phase, we expect to guarantee the security of all the routing control messages. To achieve this objectif, we perform the supervision on all of them. We detect in particular the misbehaviors discussed in the previous section. Other possible attacks are also detected for the purpose of establishing a reputation system as soon as possible.
We also suppose that the authentication of neighbors can be performed by SWAN. More details on this topic will be shown in section 5.7.

Modifications to RREQs and RREPs

We show the RREQ and RREP headers of TRP in figure 5.6 (the fields in italic are the fields of TRP that are different to SRP), an example of the route discovery of TRP in figure 5.7, and an example of direct and indirect trusts in the TRP routing in figure 5.8.
When rebroadcasting a RREQ, an intermediate node adds to the RREQ not only its identity (IP address) but also its direct trust value to its upstream node. Every node should also maintain a table to memorize the trust values it has recently attributed to each of the RREQs.
Suppose that a route is made of a series of nodes denoted S, I_1, ..., I_i, ..., I_{d-1}, D. Then, a RREQ rebroadcasted by an intermediate node I_i will be in the following form:

$$I_i \rightarrow * : IP_S, IP_D, SN, r, IP_{I_1}, ..., IP_{I_i}, CD_{I_2 \rightarrow I_1}, ..., CD_{I_i \rightarrow I_{i-1}}, h_{K_{S,D}}(IP_S|IP_D|SN|r)$$

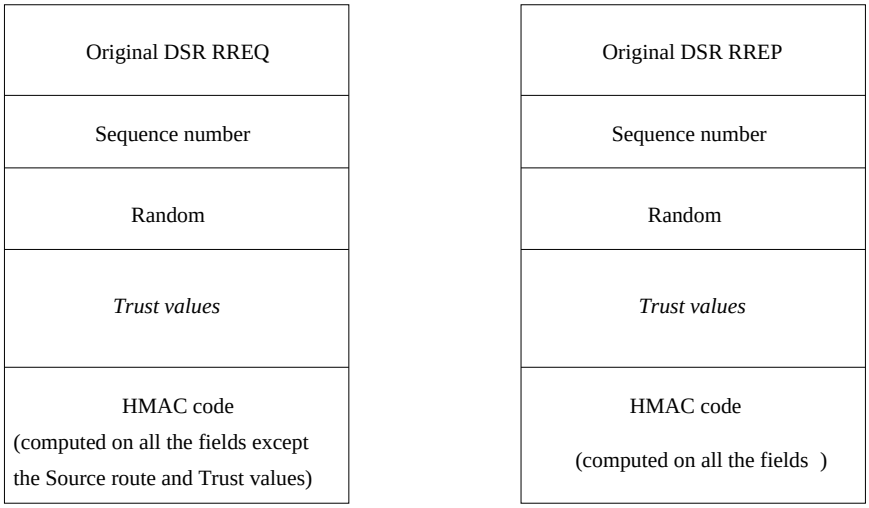
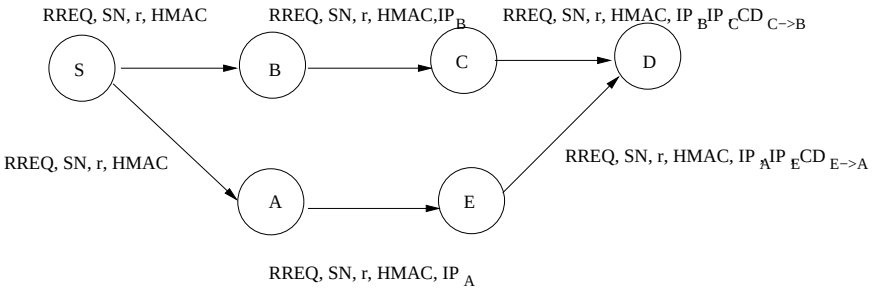
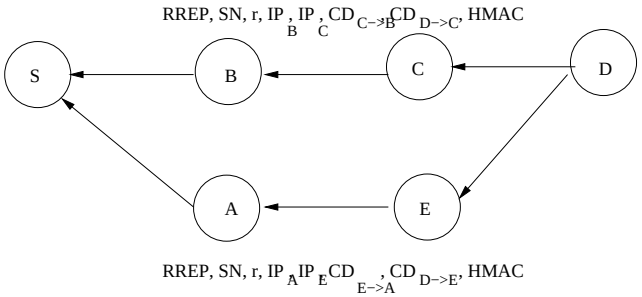


Figure 5.6: RREQ and RREP headers of TRP



(a) RREQ broadcasting example



(b) RREP sending example

Figure 5.7: Example of route discovery in TRP

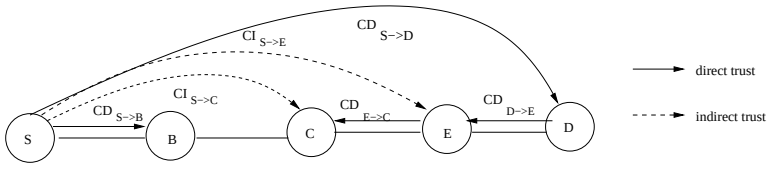


Figure 5.8: An example of direct and indirect trusts in the TRP routing

Later on, all the trust values collected in the RREQ, e.g. $CD_{I_2 \to I_1}, ..., CD_{I_i \to I_{i-1}}, ...,$ $CD_{I_{d-1} \to I_{d-2}}$, $CD_{D \to I_{d-1}}$, should be sent back to the source node within a RREP. The RREP will be prepared by the destination node in such a way that the HMAC in the RREP covers the entire RREP including the trust values.
During the forwarding of a RREP, each intermediate node I_i should check the value $CD_{I_i \to I_{i-1}}$ to ensure that $CD_{I_i \to I_{i-1}}$ has not been modified by one or more nodes among I_{i+1},...,I_{d-1}[1]. If it is modified, the packet will be silently rejected by node I_i. Therefore, during the propagation of RREQ, malicious nodes have no possibility to modify the trust values reported by the other nodes. Furthermore, during the forwarding of RREP, since the integrity of the trust values is protected by an HMAC, no modification of trust values is possible (any modification will be detected by source node).
Thus, an initiator of RREQ will obtain from each received RREP a set of unaltered direct trust values. Thanks to these values, the initiator is then able to compute indirect trusts to the nodes to which it has no direct trust value.
As defined in section 5.5.2.2, indirect trust values can be derived using direct and indirect trust values. When necessary, they may be calculated as follows:

$$CI_{S \to I_{d-1}} = \begin{cases} -1 & \text{if } CD_{S \to D} \text{ or } CD_{D \to I_{d-1}} = -1 \\ 1 - (1 - CD_{D \to I_{d-1}})^{CD_{S \to D}} & \text{otherwise} \end{cases}$$

and for $1 \leq i \leq d-2$

$$CI_{S \to I_i} = \begin{cases} -1 & \text{if } CI_{S \to I_{i+1}} \text{ or } CD_{I_{i+1} \to I_i} = -1 \\ 1 - (1 - CD_{I_{i+1} \to I_i})^{CI_{S \to I_{i+1}}} & \text{otherwise} \end{cases}$$

Optimization: In order to acquire an indirect trust to node I_i, the source node can check, before performing the above computations, whether there is a node I_j ($1 \leq i < j \leq d-1$) between I_i and D to which S has a direct trust value. If it is the case, $CI_{S \to I_i}$ can be calculated based on $CD_{S \to I_j}$ instead of $CD_{S \to D}$. As a consequence, the recommendation chain becomes shorter and the calculation of indirect trust may be more correct. The modified formulas is as follows (the computation of $CI_{S \to I_{d-1}}$ is unchanged):
For $1 \leq i \leq d-2$

$$CI_{S \to I_i} = \begin{cases} -1 & \text{if } CDI_{S \to I_{i+1}} \text{ or } CD_{I_{i+1} \to I_i} = -1 \\ 1 - (1 - CD_{I_{i+1} \to I_i})^{CDI_{S \to I_{i+1}}} & \text{otherwise} \end{cases}$$

where

$$CDI_{S \to I_{i+1}} = \begin{cases} CD_{S \to I_{i+1}} & \text{if } p^+_{S \to I_{i+1}} + p^-_{S \to I_{i+1}} \geq 1 \text{ (if } S \text{ observed } I_{i+1}) \\ CI_{S \to I_{i+1}} & \text{otherwise} \end{cases}$$

However, since I_j and S are currently not neighbors, the direct trust value $CD_{S \to I_j}$ could be out of date. In order to limit this disadvantage, a compromise can be taken between the length of the recommendation chain and the interval between the current time and the last time that S has observed I_j (this requires that the

[1] Note that this attack can also be detected by the supervision mechanism.

trust table maintains an additional field which memorizes the last time that S observed I_j). Otherwise, another compromise can be taken between the length of the recommendation chain and the number of the observations that S has made on I_j.

Blackmail attacks

Since nodes cannot modify the reputation values reported by the other nodes, the only way for an attacker I_i to introduce a forged trust value is to add a forged $CD_{I_i \to I_{i-1}}$ to a RREQ:

- If the $CD_{I_i \to I_{i-1}}$ is too low, the corresponding route trust value will also be too low. Thus, the route will not be chosen by the source node, and the attacker cannot be on an active route.
- Otherwise, $CD_{I_i \to I_{i-1}}$ can be introduced too high. However, since there is a strong probability that the trust value $CD_{I_{i+1} \to I_i}$ is low (because the neighbor I_{i+1} may have observed the misbehaviors of the attacker), and since the trust value of a route only depends on the minimum value of all the (direct or indirect) trust values, the route containing the attacker has a weak probability to have a high route trust. Thus, the route cannot be chosen.
- Even in an extreme case where all the intermediate nodes are misbehaving nodes and they all provide fake reputations, the destination (which is trusted by the source node) can still provide a bad reputation to the node I_{d-1}. Thus the route cannot be chosen.

Based on the above discussions, we can draw the conclusion that providing fake reputation into TRP is difficult. Therefore TRP is relatively robust against the blackmail attacks.
Note that each indirect trust value is only a temporary value which will influence at most one route. It will not be stored after the derivation of the corresponding route trust value. This can limit the influence of the fake recommendations.

TRP improvement

Optionally, when forwarding a RREP, every intermediate node I_i can check whether $CD_{I_i \to I_{i+1}}$ equals to -1 (in other words, whether the node I_{i+1} is not trustworthy for the node I_i). If it is the case, I_i rejects the RREP, because one untrustworthy node will make the whole route untrustworthy. This option has been implemented in an optimized version of TRP called TRP$^+$. We show in figure 5.9 the main steps of the route discovery in TRP$^+$.
With TRP$^+$, we give more authority to the intermediate nodes that we do not trust by default. Nevertheless, we estimate that this modification of the protocol cannot damage the routing security, because:

- If I_i is malicious, the route should not be chosen anyway.

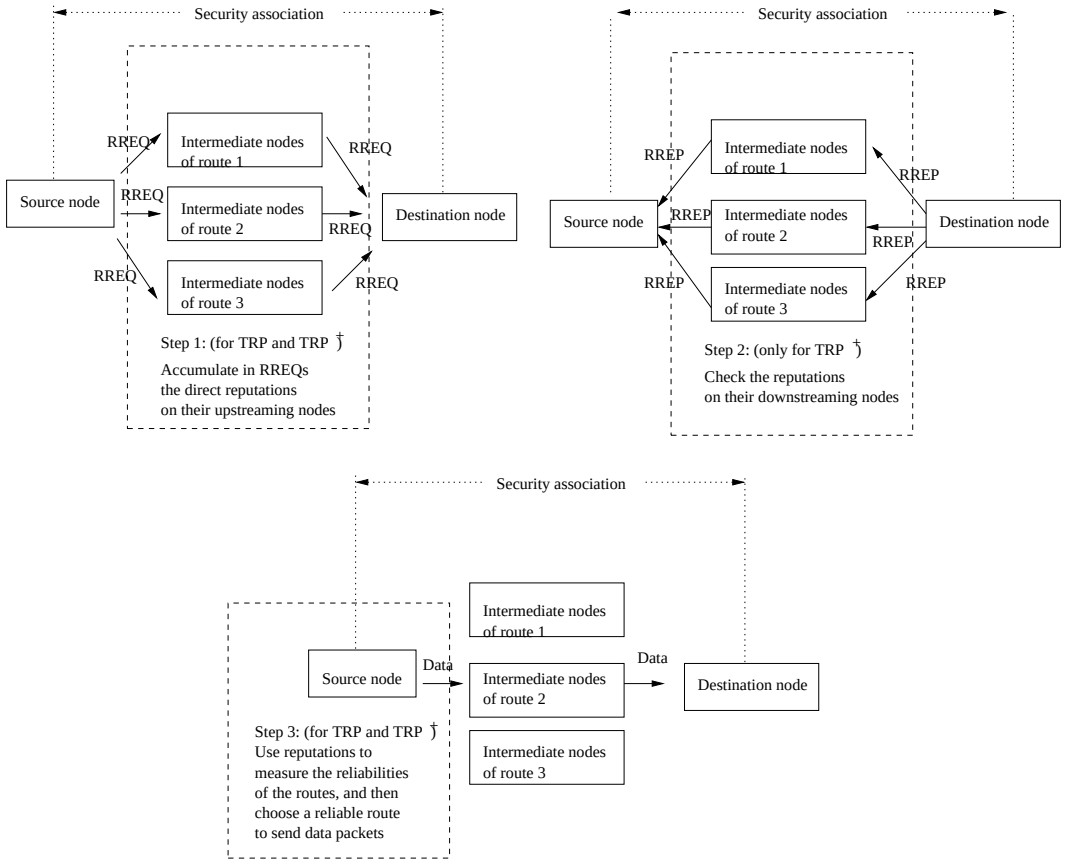


Figure 5.9: The main steps of the route discovery in TRP$^+$

- If I_i is a benign node, it should be trusted to refuse the RREP.

We can foresee that TRP$^+$ will significantly outperform TRP, since in TRP$^+$ the trust information in both directions (upstream and downstream directions) of routes is considered, while in TRP, we do only consider the upstream direction.

5.6.2.3 Route management

After a route is received and the necessary indirect trust values are calculated, the route is ready to be inserted into the route cache of the source node except in two cases: first, if there are one or more trust values equal to -1 (which means that there are untrustworthy nodes) on the route; second, if there was another route received thanks to the same RREQ which is a partial route of the current route. In the latter case, we keep the other route and reject the current route.

Then, a route can be stored in the route cache of the source node, together with its route trust value. A route trust value is computed according to the formulas defined in section 5.5.2.3, and it will represent the security level of the route. The higher is the value, the less is the possibility of encountering malicious nodes when using the corresponding route.

5.6.3 Data forwarding phase

In this section, we present our security considerations for the data forwarding phase of TRP. In section 5.6.3.1, we show in detail how routes are chosen for sending data packets. Then, in section 5.6.3.2 we discuss the route maintenance process of TRP. Finally, in section 5.6.3.3 we present the reputation management in the data forwarding phase of TRP.

5.6.3.1 Route choice

In TRP, each data packet must carry one route in its header. Thus, the source node should find out one route in its route cache for each data packet. If no route is available, it should re-initiate a routing discovery phase.
To choose a reliable route from route cache, two strategies are considered by us:

- Always choose the route with the highest trust value, regardless of the lengths of the available routes. However, if there are multiple routes which have the highest trust value, we will choose the shortest route among them.
- Set at first a threshold as the lowest acceptable trust level, and then choose the shortest route among all routes having a trust level equal or greater than the threshold. If there is no such a route available, we may give up the data sending or re-initiate a new RREQ. The value of the threshold may be evaluated according to the time.

The first strategy emphasizes the security, while the second strategy is a compromise between the security and the performance.

5.6.3.2 Route maintenance

In section 5.6.2.1, we have discussed some flaws of SRP in the routing discovery phase. Here, we show a flaw of SRP in the route maintenance phase (with RERR messages).
Since the RERR messages are not authenticated (due to the fact that the initiators of RERRs have no key to authenticate themselves to source nodes) in SRP, malicious nodes can invalidate the available routes by sending forged RERRs with spoofing. A fast moving attacker knowing the topology of the network can thus invalidate a lot of routes.
The utilization of NLP can in partial resolve the problem, since with NLP the attacker can attack only when it can spoof both the MAC and the IP addresses of a node on route. However, NLP is not sufficient to prevent the attack. Even if the neighbor relationships are known to each node, it still be difficult to detect the attack, since an attacker can place itself close but out of the transmission range of the node spoofed.
In TRP, this attack is avoided by adding an authentication mechanism to the RERR messages. If the initiator of a RERR message has a shared key with the source node, it should use it to compute an HMAC for the authentication of the RERR. Otherwise, since a PKI system is also required for establishing the security associations

between nodes, a signature can be used to sign the RERR. Moreover, this attack can also be detected by SWAN.

5.6.3.3 Reputation management in the data forwarding phase

For any data packet sent or forwarded, the sender or the forwarder, namely S or I_i ($1 \leq i \leq d-2$), saves a copy of the packet in its supervision buffer (when SWAN is not used). Then it supervises the action of the next node, I_1 or I_{i+1}, to check whether or not the latter correctly executes the forwarding function.
If I_1 or I_{i+1} correctly forwards the packet within a limited time period, S or I_i increments the value of $p^+_{S\rightarrow I_1}$ or $p^+_{I_i\rightarrow I_{i+1}}$. Otherwise, $p^-_{S\rightarrow I_1}$ or $p^-_{I_i\rightarrow I_{i+1}}$ will be increased. The value of $CD_{S\rightarrow I_1}$ or $CD_{I_i\rightarrow I_{i+1}}$ will thus be updated.
If multiple data packets are found not well forwarded, a RERR message must be sent back to the source node, and the rating on the next node is greatly decreased. A highly mobile node may thus have a low reputation, since it can be included in a route but leaves the route due to mobility. However, since the routes depending on it are not stable, the node is not suitable to be used in the ad hoc routing as intermediate node.

5.6.4 Residual vulnerabilities

In figure 5.10, we show the threat tree (c.f. chapter 2) of TRP. In the tree, we mention, under each misbehavior taken into consideration, the possibility to avoid or detect it in TRP.
Through the tree, we note that even though TRP permits to counter a large number of ad hoc routing misbehaviors, the following weaknesses (usually except the hypotheses) still reside in TRP:

- *Sending forged RERR:* Sending a forged RERR can invalidate a functional route. In TRP, this attack is not detectable due to the limited supervision (since with SURO only the nodes sending RERRs care whether there are broken links in their downstream directions). CORE has the same problem.

  Solution: We can use the supervision in neighborhood to detect this attack, at the price of some additional overhead.

- *Selfish routing behavior:* In TRP, we are only able to detect the selfish forwarding nodes, but not the selfish routing nodes. Moreover, in oder to save energy, an intermediate node can anyway provide -1 as its trust value to its upstream node.

  Solution: Each node can periodically decrease its reputation to each of the nodes in its *reputation table* which seems distant. Even though a distant node (maybe due to the mobility) can also be punished in this way, we think that it is reasonable to less trust a node that we have recently not cooperated with. Unfortunately, we have no solution until now for the selfish nodes which provide low reputations.

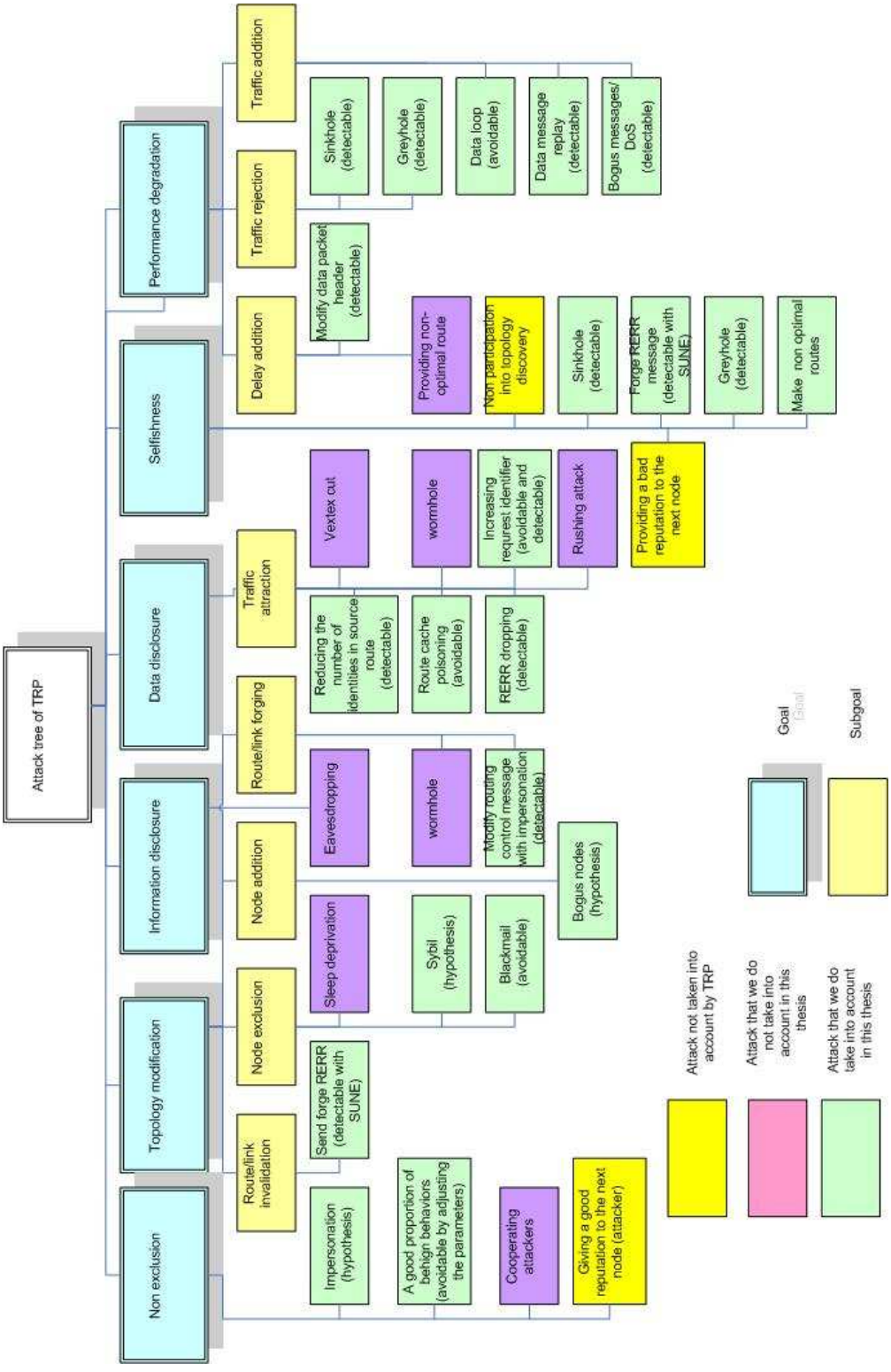


Figure 5.10: Threat tree of TRP

- *Giving a good reputation to the next node.* We have already discussed in section 5.6.2.2 that such an attack cannot achieve its objective. However, the analysis is based on the hypothesis that the node also misbehaves elsewhere, and will thus have a bad reputation itself.

Finally, like all the protocols using a reputation system, TRP needs a *training phase* before it can be really security-efficient. This phase is used to detect the misbehaving nodes and to establish a relative stable reputation system. The duration of the phase can partially be adjusted with our model parameters, say α and β.

5.6.5 Comparison with other protocols

In comparison with other approaches using a similar reputation system, such as CORE [MM02] and CONFIDANT [BB02a], TRP has the advantage of protecting its reputation exchanges against modifications and the blackmail attacks without additional security mechanism. Moreover, TRP combine more tightly the routing and the reputation system together.
However, compared to CONFIDANT, TRP has a main disadvantage which is that the training phase would be longer.
In comparison with the SMT [PZ03] protocol (c.f. section 3.4.2.3) which transfers data on multiple routes, TRP has the disadvantage of needing a training phase before becoming operational. However, it has the advantages of not requiring additional messages (Ack messages are required in SMT) and isolating misbehaving nodes.

5.7 TRP with SWAN

Since the traditional watchdog technique has several issues in terms of authentication and memory efficiency, in the previous chapter we suggested a secured watchdog technique called SWAN. It can guarantee to some degree the efficiency and the correctness of supervision systems. In this section, we show how SWAN can be applied to TRP and the improvements that SWAN can bring to TRP.

5.7.1 Scheme

In TRP, the fixed fields in RREQ and RREP are already protected by an HMAC code. However, since only end nodes are able to verify the original HMAC, a new HMAC is required to provide the authentication and the integrity check to intermediate nodes during the supervision. Note that neither source node nor destination node adds the new HMAC.
Suppose that at time interval τ_k, a node I_i rebroadcasts a RREQ which will be received by node I_{i+1}. Let

$$M\_Fix =< IP_S, IP_D, SN, r, h_{K_{S,D}}(IP_S|IP_D|SN|r) >$$

$$M\_Var =< IP_{I_1}, ..., IP_{I_i}, C_{I_2 \to I_1}, ..., C_{I_i \to I_{i-1}} >$$

we then have

$$I_i \rightarrow * :< IP_S, IP_D, SN, r, IP_{I_1}, ..., IP_{I_i}, C_{I_2 \rightarrow I_1},$$

$$..., C_{I_i \rightarrow I_{i-1}}, h_{K_{S,D}}(IP_S|IP_D|SN|r), h_{h^{n-k+1}(s_i)}(h(M\_Fix)|M\_Var) >$$

I_i stores $< IP_S, r >$ as packet's identity, $h(M\_Fix|M\_Var)$ as the hash, and $< IP_{I_1}, ..., IP_{I_i}, C_{I_2 \rightarrow I_1}, ..., C_{I_i \rightarrow I_{i-1}} >$ as variable fields.
Upon receiving the packet, I_{i+1} should add to it its identity and its trust value on I_i before rebroadcasting it:

$$I_{i+1} \rightarrow * :< IP_S, IP_D, SN, r, IP_{I_1}, ..., IP_{I_i}, IP_{I_{i+1}}, C_{I_2 \rightarrow I_1}, ..., C_{I_i \rightarrow I_{i-1}}, C_{I_{i+1} \rightarrow I_i},$$

$$h_{K_{S,D}}(IP_S|IP_D|SN|r), h_{h^{n-j+1}(s_{i+1})}(h(M'\_Fix)|M'\_Var) >$$

where

$$M'\_Fix = M\_Fix =< IP_S, IP_D, SN, r, h_{K_{S,D}}(IP_S, IP_D, SN, r) >$$

$$M'\_Var =< IP_{I_1}, ..., IP_{I_i}, IP_{i+1}, C_{I_2 \rightarrow I_1}, ..., C_{I_i \rightarrow I_{i-1}}, C_{I_{i+1} \rightarrow I_i} >$$

I_i observes the message and identifies the message. It further checks I_{i+1} and $C_{I_{i+1} \rightarrow I_i}$ to see whether they are respectively a valid IP address and a valid trust value. Finally, it checks $h(M\_Fix)$. For the future authentication, it stores $IP_{I_{i+1}}, C_{I_{i+1} \rightarrow I_i}$ and j.
During the next time interval, upon receiving the key $h^{n-j+1}(s_{I_{i+1}})$, I_i checks the validity of $h^{n-j+1}(s_{I_{i+1}})$ by computing j hashes:

$$IP_{i+1} =< \text{IPv6 prefix}, \text{hash-64}(h^{j-1}(h^{n-j+1}(s_{I_{i+1}}))) >$$

The number of the hash operations can also be reduced. For example, if the key in the previous interval $h^{n-j+2}(s_{I_{i+1}})$ is already known, then only one hash operation will be sufficient:

$$h^{n-j+2}(s_{I_{i+1}})) = h(h^{n-j+1}(s_{I_{i+1}}))$$

If the check is successful, I_i verifies in addition whether $h_{h^{n-j+1}(s_{i+1})}(h(M'\_Fix)|M'\_Var)$ is valid.
The other types of messages, such as data, RREP and RERR, do not change their contents during their forwarding. Therefore, for those messages denoted M, M_Fix equals to M and M_Var equals to null, and the authentication of them can follow exactly the same process as described in section 4.4.2.

5.7.2 Security and performance improvements

SWAN turns impersonation impossible and misbehavior detection more certain. Thanks to the authentication, bad affection of reputations becomes difficult to realize. In figure 5.11, we show the threat tree of TRP with SWAN, wherein we dyed the terms that are improved by SWAN blue.
In addition, with SWAN an intermediate node denoted I_i can detect the impersonation attacks committed by its next nodes by using a mechanism similar to Ariadne

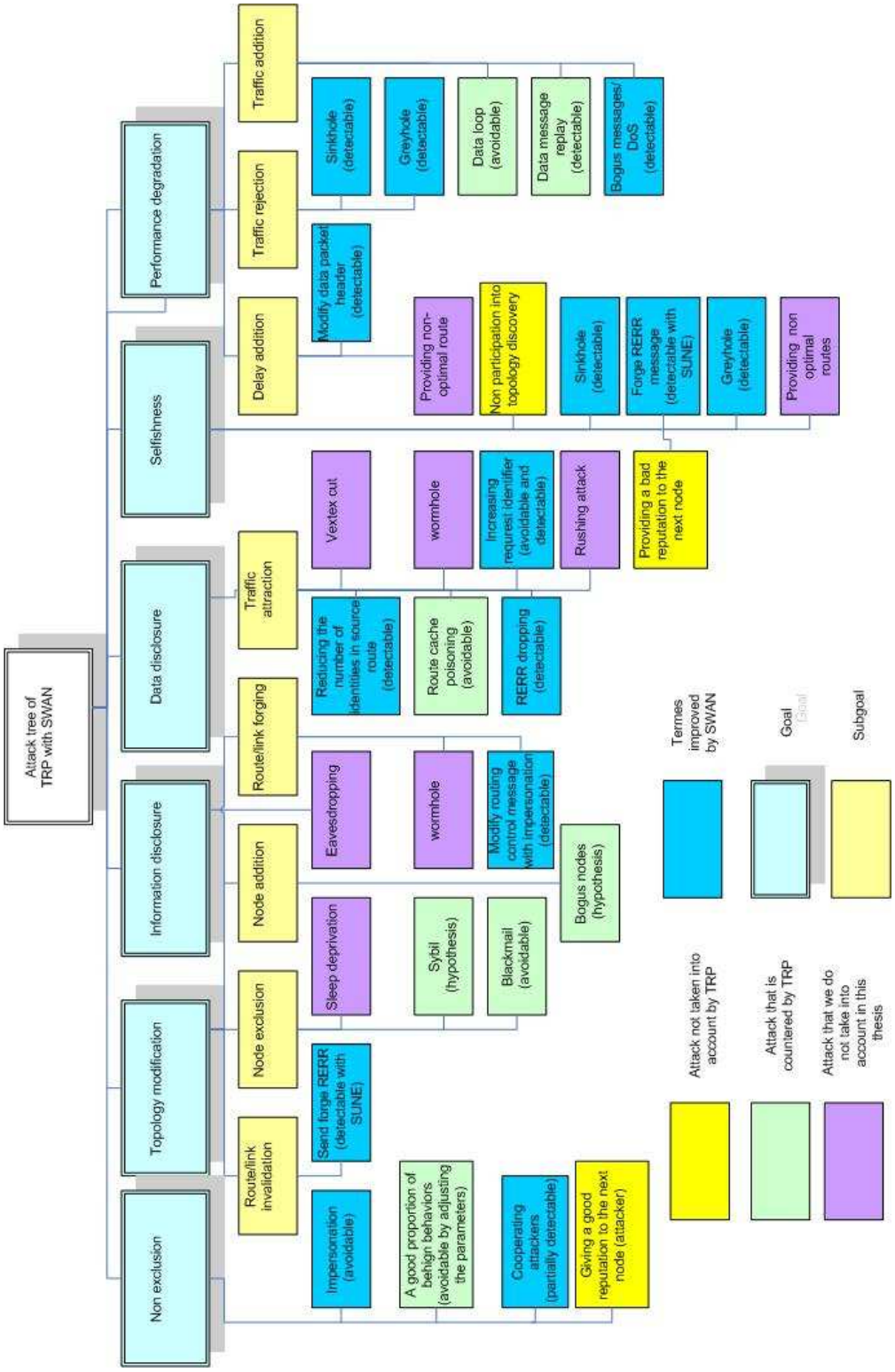


Figure 5.11: Threat tree of TRP with SWAN

[HPJ02] (c.f. section 3.4.2.1). If a RREQ forwarded by a node declared as I_{i+1} cannot be authenticated, then a RREP sending back a route $S, ..., I_i, I_{i+1}, ..., D$ can be rejected by node I_i. This measure can further improve the security of TRP.
Another improvement of SWAN is the reduction of storage requirement in the supervision system without loss in the observation capacity.

5.8 Simulation

In this section, we study via simulations the security performance of TRP and its variations, say TRP$^+$ and TRP with SWAN (we call it TRPS later on).

5.8.1 Implementation of the protocols

Our simulations are carried out under the network simulator NS-2 [pro98]. The initiation of the protocols, especially the initiation of the security parameters and keys, are done at the beginning of each simulation. The HMAC function, the generation of the keys and the random seeds, etc..., are realized by using an external library, e.g. the OpenSSL [CEH$^+$] cryptography library named *crypto*.
However, we found that calling an external library will not permit to simulate the computational delay caused by the cryptographic operations, but will increase the duration of each simulation execution. Thus, to take the delay of cryptography into consideration and to reduce the simulation execution time, we can use an alternative method instead of calling a cryptography library: we can set a timer for each cryptographic operation executed in the simulations. In order to simulate the delay of cryptography, the duration of each timer will be equal to the execution time of the corresponding cryptographic operation. We can refer to an authoritative benchmark to know the value of timers.
TRP is implemented on the DSR module that is already integrated in NS-2. To start with a primary DSR version without many optimizations, we have chosen the NS-2 version NS2.1b7a.
In TRP we choose the *mobicache* as the route cache implementation, since in TRP each route should be stored separately with a route trust value. We do not prefer the *link cache* implementation even though it is more efficient in terms of storage, since it only stores the connectivities between nodes (links), and routes are calculated on-demand.
When a new packet is to be buffered in the promiscuous buffer, the oldest packet in the buffer will be dropped if the buffer is already full. Each TRP buffer entry can be used to save a packet.
Each TRPS buffer entry is in the following format: $< t, UID, ADsender{-}64, h(M\_Fix), h_{key}(h(M\_Fix)|M\_Var), supervised, authenticated, M'\_Var >$, where *supervised* and *authenticated* are two flags marking the states of the entry, and t is the timestamp registering the time that the entry is buffered. We do not save the prefix of the IP addresses since it is the same for every IP address.

Parameter	Value
Simulation time	10,000s for TRP and TRP$^+$, and 1,000s for TRPS
Field range	700m × 700m
Number of nodes	25
Number of attackers	1 for TRP and TRP$^+$, and 20% for TRPS
Propagation model	Two-way ground
Power range	250m
Mobility model	Random way point
Mobility	Low - 100s as pause time and 2m/s as maximum speed Medium - 20s as pause time and 5m/s as maximum speed High - 5s as pause time and 20m/s as maximum speed
MAC protocol	IEEE802.11
MAC queue size	50
Traffic type	FTP CBR 2 pkt/s
Number of flow	22
Packet size	512 bits

Table 5.1: Simulation model and parameters

In order to periodically refresh trust levels of routes, each route is set a timeout. Once the timeout of a route is reached, the route is removed from the route cache. This can guarantee the freshness of both routes and their trust values.

5.8.2 Implementation of misbehaviors

Multiple misbehaviors are implemented and tested under TRP, TRP$^+$ and TRPS, such as non forwarding of data packets (also called selfish forwarding behaviors or sinkhole), partial forwarding of data packets (also called greyhole), introduction of loops into RREQs, modification of routes in RREQs, modification of data packets' headers, generation of RERRs with spoofing, etc. In our simulations, all the above misbehaviors are shown as detectable by the watchdog.
For TRP and TRP$^+$, we present our simulation results with the following attack: an attacker modifies a packet before forwarding it; moreover, in order to keep on attacking the attacker will never initiate or forward a RERR whether or not there is a broken link in its downstream direction.
To compare TRPS with TRP, we test the following attack: each attacker observes whether there is any data flow passing through its neighborhood; if so, it spoofs the address of the neighbor node that should forward the flow, and then sends wrong packets.

5.8.3 Simulation configuration

Tables 5.1 shows the parameters used in our simulations. To simulate TRP and TRP$^+$, the network that we simulated contains 25 nodes, among them 24 nodes are

Protocol(s)	Parameter	Value
TRP and TRP$^+$	α	0.75
	β	10
	Promiscuous buffer size	20
TRPS	Δt	4s
	T_Max	1,000s
	$T0$	0s

Table 5.2: Setting of the protocols

benign and one node is malicious. To simulate TRPS, we randomly choose 20% of the nodes as attackers.
Under the random waypoint mobility model, three mobility scenarios are tested:

- low mobility - 100s as pause time and 2m/s as maximum speed.
- medium mobility - 20s as pause time and 5m/s as maximum speed.
- high mobility - 5s as pause time and 20m/s as maximum speed.

FTP is used as application protocol with 22 random Constant Bit Rate (CBR) sources and a packet rate of 2 packets per second. The simulation time is set to 10,000 seconds and the simulation area is a square of 700 meters.
Figure 5.2 shows the parameters used by the protocols. As the reputation parameters of TRP, α is set to 0.75, and β is set to 10. The promiscuous buffer size is set to 20 (each node can simultaneously save a maximum of 20 packets waiting for supervision). For TRPS, Δt is set to 4 seconds, T_Max is $1,000$ seconds, and the starting time $T0$ is set to 0 (thus each Hash chain contains 251 hash values).

5.8.4 Measures

First, for TRP and TRP$^+$ we can measure the average direct trust value on the misbehaving node. Suppose that n_1, ..., n_p denote benign nodes, and X_1, ..., X_{N-p} denote misbehaving nodes, we can compute the average direct trust value on the misbehaving nodes as follows:

$$\text{Average trust value on misbehaving nodes} = \frac{\sum_{i=1}^{p}\sum_{j=1}^{N-p} CD_{n_i \to X_j}}{p \times (N-p)}$$

This measure allows us to see the evolution of direct trust values on misbehaving nodes, which is a useful way to verify the effectiveness of our supervision system.
Otherwise, we can also measure the average direct trust value on benign nodes in TRPS, by using the following formula:

$$\text{Average trust value on benign nodes} = \frac{\sum_{i=1}^{p}\sum_{j=1}^{p} CD_{n_i \to n_j}}{p \times (N-p)}$$

Second, we measure the number of data and control packets that are attacked or dropped during the ad hoc routing. This measure will allow us to check the security efficiency of TRP.
Third, attention is paid to the routing performance. For this, we measured the average route length, routing overhead, average end-to-end delay of data packets, and the total storage overhead of watchdog.

5.8.5 Simulation results

5.8.5.1 Security results

Figure 5.12 shows the average direct trust value on misbehaving nodes in TRP under the three mobility scenarios. We can see from the figure that, regardless of the scenarios, any average trust value on misbehaving nodes starts from 0 and decreases as time progresses. If it drops fast at the beginning, it drops slower later, and vice versa. This phenomenon can be explained as follows. If misbehaving nodes have the possibility to misbehave from the beginning, their reputations will be decreased rapidly. Then, having low reputations will prevent them from keeping on misbehaving, thus their reputations will decrease slower later.
We also note that in most cases, the average trust is rather stable during the last half of the network time. We believe that it is due to the fact that misbehaving nodes have already been found misbehaving by a lot of nodes. Then, thanks to our secure routing mechanism, not many misbehaviors can further be committed by them. Therefore the supervision has less possibility to discover the nodes, and the average trust has less possibility to decrease.
Finally, we found that all the average trust values are always larger than -1, thus we can draw the conclusion that it is not easy to make misbehaving nodes be directly recognized by all nodes in the network. This can be explained with the two following reasons. First, with a random mobility, not all nodes have the possibility to be neighbors of the misbehaving nodes. Second, as described in the previous paragraph, if misbehaving nodes can no more misbehave, their reputations will no more decrease. Therefore we can justify the necessity to define the indirect trust.
Figure 5.13 shows the average direct trust value on the benign nodes in TRP and in TRPS under the high mobility scenario. We found in this figure that TRPS can help to avoid the spoofing attack. That is to say that reputations of benign nodes will not be badly affected by the spoofing attackers in TRPS.

Figures 5.14, 5.15 and 5.16 show, respectively in the low, medium and high mobilities, the number of misbehaviors that have taken place in the whole network. The three curves in the three figures represent respectively the case of SRP, TRP and TRP$^+$.
We found in these figures that the number of misbehaviors in TRP or TRP$^+$ is always smaller than that of SRP regardless of the scenarios. This is normal, since compared to SRP, TRP and TRP$^+$ employ additional security mechanisms.
We also found that the differences between the three protocols become larger as

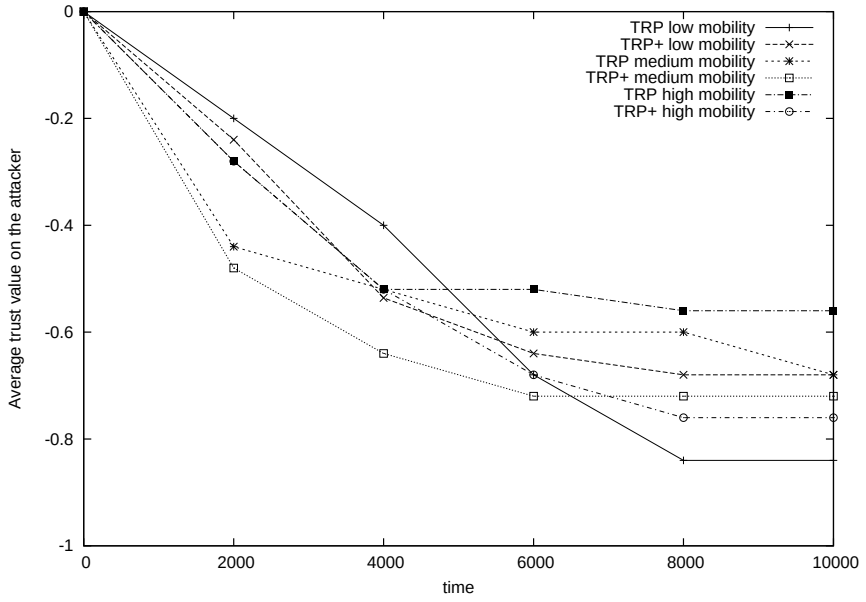


Figure 5.12: Average trust value on the misbehaving node

time progresses. This shows that our reputation system can take effect after the reputations are well established. In the meantime, TRP$^+$ largely outperforms TRP, since in TRP$^+$ the reputation system is better used than in TRP. We check in TRP only the reputations on the upstream direction of routes, while in TRP$^+$ the reputations on both directions are checked. Note also that it is more efficient to verify the downstream reputations of routes, since downstream is the direction of data forwarding. As a conclusion, we believe that TRP is more efficient than SRP, and TRP$^+$ is still more efficient than TRP.

With the figures, it is also found that the total number of misbehaviors can be stabilized. We believe that it is because that, as time progresses, misbehaving nodes will have less and less possibility to be included into routes and to attack. In other words, attackers will be excluded from the ad hoc routing. Thus, the security objective of TRP is achieved.

Finally, as expected, we found that the stronger is the mobility, the better are the security results. We believe that this is due to the fact that a dynamic network topology can help nodes to encounter each other, thus to discover the misbehaving nodes more easily. This feature is quite interesting, since the mobility usually plays a negative role for MANET security.

5.8.5.2 Performance results

We compared the performance of our protocols with that of the DSR protocol.

We can observe in figure 5.17 that, regardless of the scenarios, the average route length in TRP or in TRP$^+$ increases less than 3% compared to that of DSR. This increment is not significant.

Figure 5.18 shows the end-to-end delay of TRP under different mobility scenarios

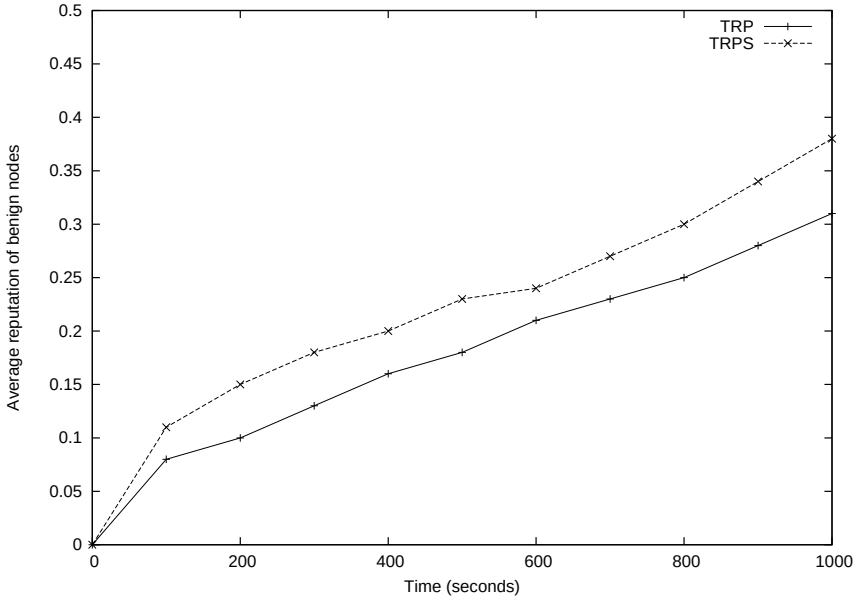


Figure 5.13: Average trust value on benign nodes in TRP and TRPS

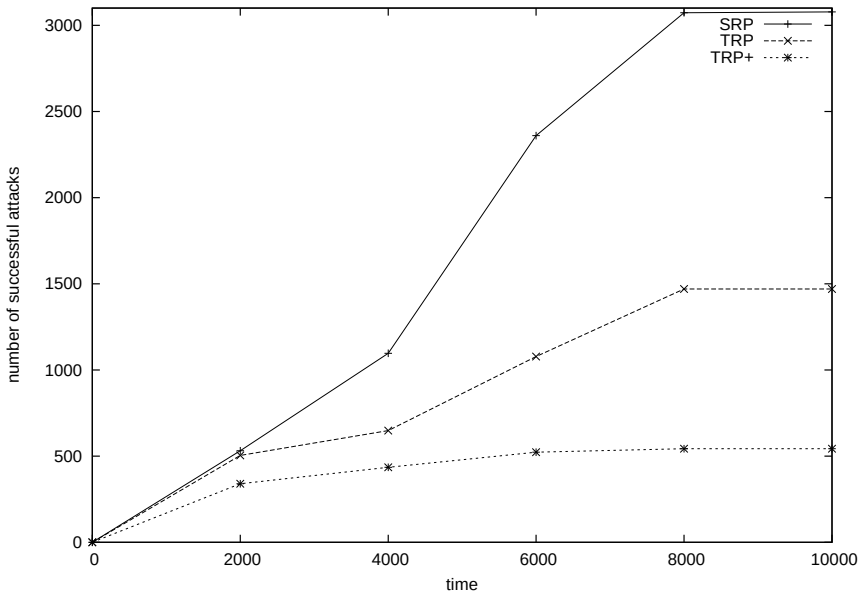


Figure 5.14: Misbehaviors: low mobility

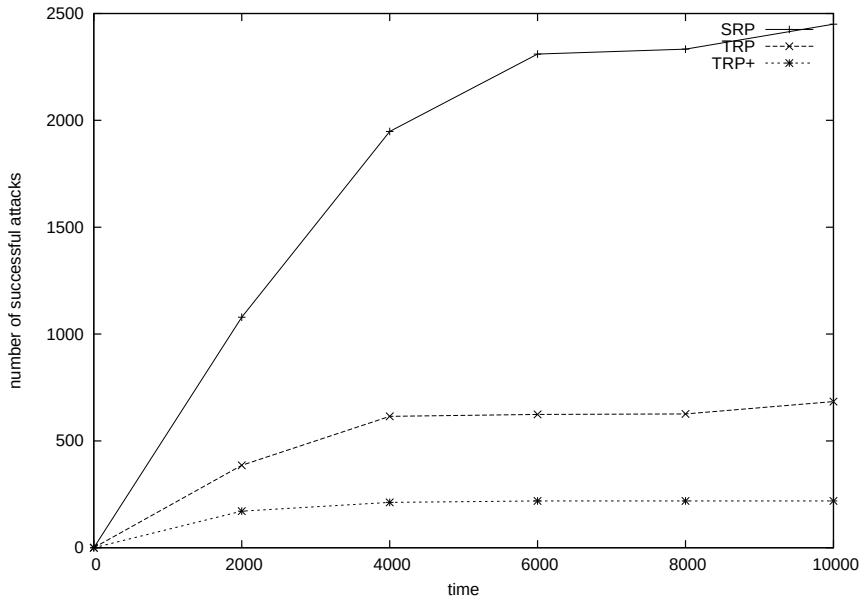


Figure 5.15: Misbehaviors: medium mobility

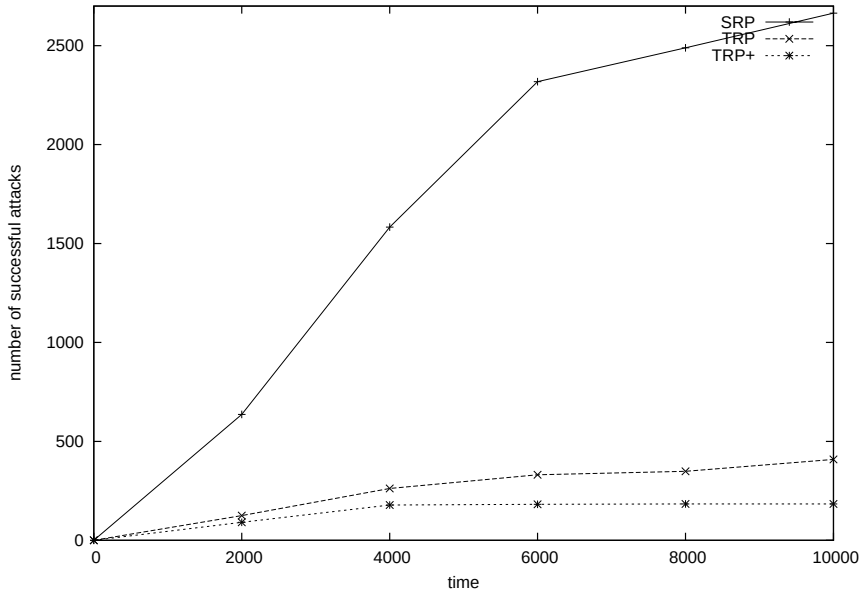


Figure 5.16: Misbehaviors: high mobility

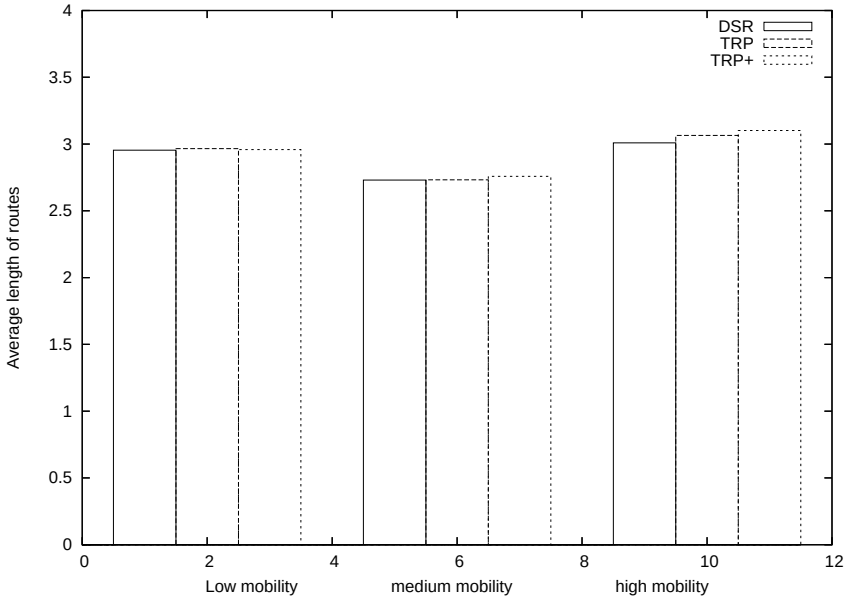


Figure 5.17: The average route length in DSR, TRP and TRP$^+$

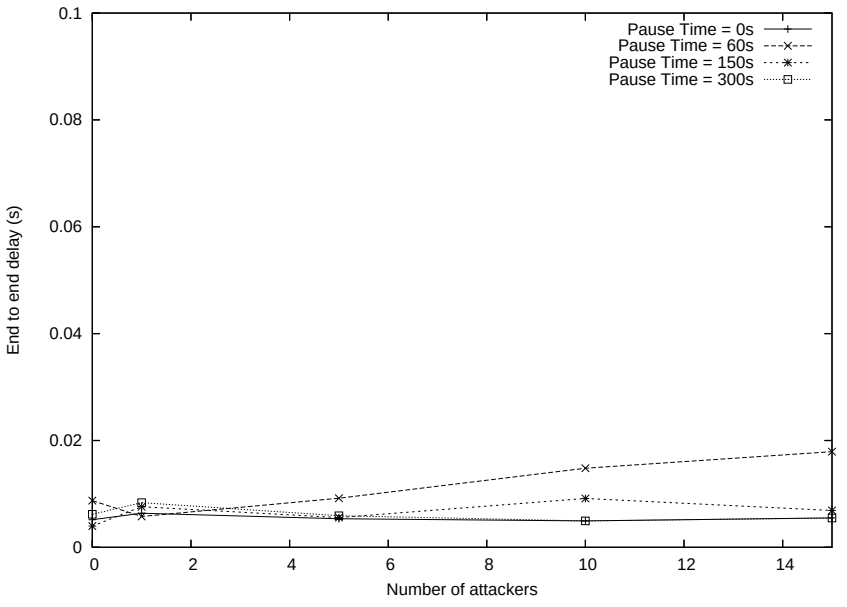


Figure 5.18: The end-to-end delay of TRP under different mobility scenarios

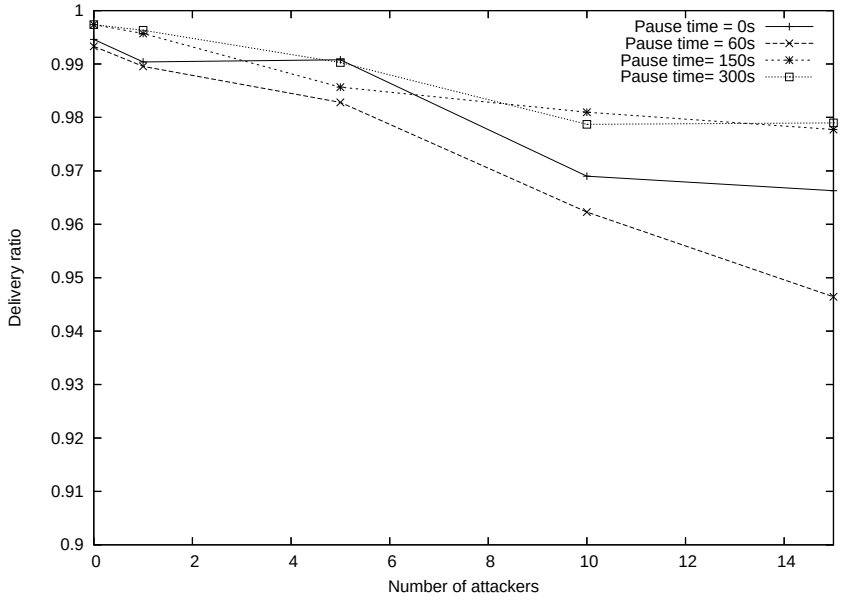


Figure 5.19: The delivery ratio of TRP under different mobility scenarios

with until 15 attackers in a 50-node network, and figure 5.19 shows the corresponding delivery ratio.

In terms of communication overhead, in TRP or in TRP$^+$ no new message is added but the sizes of RREQs and RREPs are increased due to the addition of the new header which is used to transport trust values. However, the routing overload is considerably increased compared to DSR, since many performance optimizations of DSR are removed by us. We consider it the most important price of the security in TRP.

We also measured the end-to-end delay and the routing overhead of TRPS. We found that the average end-to-end delay is not varied compared to TRP, since in SWAN nothing including cryptographic operations can influence the delay of data sending. As for the routing overhead, the additional KD messages represent about 19% of the total number of network packets. But since the tested traffic has a low rate of 2 packets/s, we believe that this percentage will drop when we increase the packet rate.

Finally, figure 5.20 shows the advantage of SWAN in terms of storage overhead (we only store the IP header and data). It compares the case of TRP (the case of TRP$^+$ will be the same as that of TRP) to the case of TRPS. We can see that the gain of SWAN is about 50%.

5.9 Conclusion

In this chapter, we proposed a secure reactive routing protocol named Trust-based Routing Protocol (TRP) for ad hoc networks. TRP is based on the source routing algorithm and a reputation system. It uses HMAC to protect routing control mes-

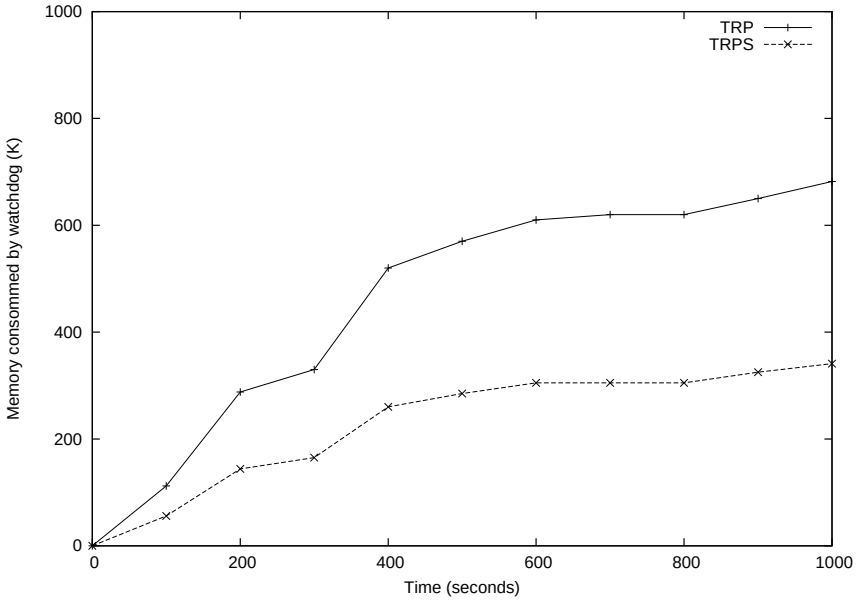


Figure 5.20: Watchdog storage requirements in TRP (TRP$^+$) and TRPS

sages, and uses the reputation system to rate routes. Then, based on the ratings of routes, source nodes can choose the most reliable routes to send their data packets.

The most important particularities of TRP are the reactive reputation exchange and the integration of reputation exchange into routing. In fact, the reputation exchange in TRP is achieved by using the DSR routing messages. This method has the following three advantages. First, the reputation exchange can be done on-demand only when the security level of some routes are needed to be measured. Second, the exchange can be done only with the routes that may be used in the routing. Third, even though the second-hand reputations are sometimes used by TRP, there is no additional packet used for the transportation of reputations. Therefore the integration of reputation exchange into routing can contribute to the reduction of the overhead caused by the use of reputation system. Moreover, thanks to the design of the routing scheme, TRP is also relatively robust to the blackmail attacks.
Our simulations showed that TRP is able to fight against a large number of ad hoc attacks during both the topology discovery phase and the data forwarding phase. Furthermore, a variation of TRP, TRP$^+$, which can take better advantage of the reputation system, outperforms TRP in terms of security.
TRP can also be improved by SWAN, because the latter is able to provide the security and some performance improvements to the watchdog which is the base of the reputation system. Some simulations showed that TRPS (TRP with SWAN) can achieve its objectives.
However, due to the training phase that is required to establish the reputations, TRP is more suited to the MANETs having a long lifetime and a dynamic topology.

In the next chapter, we will propose some security approaches for a proactive ad

hoc routing protocol, namely OLSR.

Chapter 6

HPLS and TCSec: Securing OLSR

"It takes two to make a quarrel."

– Han Fei Zi (about 280 B.C. - 233 B.C.)

6.1 Introduction

Due to the quantity of information to be secured, reactive protocols are usually considered easier to secure than proactive ones. This can be explained as follows:

- Reactive ad hoc routing protocols often exchange less control messages than proactive ones. That is to say that only information about a limited part of network topology (some routes) is exchanged on-demand. Besides, the routes are needed to be secured only before they will be used in the ad hoc routing. Unfortunately, to secure proactive protocols, we need to secure continuously the *whole network topology.*

- To secure routes in reactive protocols, it is sufficient to authenticate each node on the routes, ensure the authenticity of the routes, and ensure that the intermediate nodes are not misbehaving. However, to secure the whole network topology in proactive routing protocols, we need to guarantee the authenticity and the authentication of every topology information entry contained in every routing message, and also exclude misbehaving nodes from the routing.

Nevertheless, we believe that proactive routing protocols are worth being secured, because:

- Differently to the reactive protocols, they have their advantages and their own applications (c.f. section 2.2.2). For example, they can be used in some real-time applications since they can provide a short routing delay.

- When a reactive routing protocol is secured, its performance advantage compared to proactive protocols is less obvious, especially when the ad hoc network is highly mobile and has heavy traffic.

Moreover, proactive protocols have the following advantages which can help the design of security mechanisms:

- Since proactive routing messages are sent periodically, there could be relationship between the successive messages.
- As analyzed in section 3.5.2, it is easier to prevent selfish behaviors within proactive routing protocols.

In this chapter, we study the security of the OLSR protocol [CJ03] (c.f. section 2.2.2.2), which is indeed the first standardized MANET proactive routing protocol. We do not study the security of DSDV since it is already replaced by the AODV protocol which is reactive.
OLSR is a proactive link state routing protocol based on OSPF. Differently to the classical link state routing, it uses the MPR technique to reduce the routing overhead caused by pure flooding. It is especially suitable for large and dense ad hoc networks. However, OLSR is vulnerable to malicious attacks and selfish behaviors, and a complete description of the OLSR security issues can be found in [CB05] or in section 2.7.
Currently two main secure mechanisms[1] exist for OLSR, respectively *OLSR signature message* [ACL+05] and *ADVSIG* (c.f. section 3.4.3.3) [RACM04].
The first solution, *OLSR signature message*, secures OLSR by means of adding a timestamp and a signature to each routing control message. It can guarantee the authentication and integrity of the routing messages such as TC and HELLO. Therefore, it can prevent external attackers from forging false routing information. However, it cannot prevent compromised nodes from forging and propagating messages containing fake routing information. As a consequence, the routing can still be misled.
The second solution, ADVSIG [RACM04], has improved the security of *OLSR signature message* against compromised nodes. This is achieved by appending a new header to the messages of *OLSR signature message*. The new header contains multiple signatures, for the purpose of permitting each link information entry to be confirmed by the two ends of the link. Thus, ADVSIG guarantees not only the

[1] Other mechanisms designed for OLSR can be found, for example, in [WHiKlS05], where it is discussed the effect of replay attacks on OLSR and a secure scheme based on Message Sequence Number (MSN). Each node maintains the MSNs of their most recently received HELLO messages, and upon receiving new HELLO messages, the new MSNs are compared with the stored MSNs for the freshness check.

In [WlSiK05], each OLSR node will maintain two routing tables, which are respectively a *trusted routing table* containing only trusted nodes, and an *ordinary routing table* containing ordinary nodes. When sending data, it is up to source node to choose which routing table should be used.

In [HHF05], the authors apply the wormhole detection mechanism and the authentication to strengthen the neighbor establishment of OLSR. It uses digital signature to protect the routing packets and Hash chain to protect TTL and HC.

However, the above propositions only treat some aspects of the security problems of OLSR. They do not secure OLSR routing protocol as a whole like done by OLSR signature message and ADVSIG.

authentication and the integrity of routing messages, but also the authenticity of the routing information.
Nevertheless, for low capacity nodes, the energy consumption, the additional routing overhead and the computational delay caused by the multiple signatures in ADVSIG can be significant due to the intensive computation, the important length of multiple signatures in headers and the cryptographic computational time. This will be confirmed by the simulation results presented in section 6.4.4.
In this chapter, we first introduce a security flaw in ADVSIG (which is also presented in the new version of Raffo's thesis [Raf05]) and a solution to improve it. We then present and evaluate two lightweight mechanisms to secure OLSR, named respectively *Hash Proved Link State* (HPLS) and *Secured TC* (TCSec). We require that these new mechanisms have a slightly better security level than ADVSIG, and can avoid the excessive security overhead brought by ADVSIG.
The coherence check is used by both mechanisms. In HPLS, we adopt the idea of *proof* (c.f. section 3.4.3.3) which is introduced by ADVSIG, but we use Hash chains instead of digital signatures to reduce the cryptographic overhead. In TCSec, after appending an addition header which contains the *MPR set* to each TC message, the most of signatures are replaced by coherence check between TC messages. The simulation results show that the approaches are both lightweight and robust in ensuring the authentic topology discovery in OLSR.
The rest of the chapter is organized as follows. In the first place, we introduce the notations that are used in the chapter in section 6.2. In section 6.3, we point out a flaw in ADVSIG, and then propose an improvement of ADVSIG to resolve the flaw. In section 6.4, we detail our propositions for securing OLSR (simulation results are shown in subsection 6.4.4). Finally, we present some discussions in section 6.5, and we conclude the chapter with section 6.6.

6.2 Notations

In the following, we list the notations that are used in this chapter in their appearing order.

Notation	Meaning
A, B, C, E	node
λ	a link state
λ^p	the link state previous to λ
ϕ	no *proof* or *certificate* possible
$< M >_{SK_A}$	the message M signed by the private key of node A
t_i	the ith timestamp
$T_A(t_i)$	timestamp at local time t_i of node A
M	a message
$< \{"A : state"\}, \ T_B(t_i) >_{SK_B}$	a *proof* or a *certificate* signed by node B showing that at time t_i of node B, B has a *state* link with node A
$A \rightarrow \mathcal{N}(B, C, ...) : M$	a node A broadcasts the message M to its direct

	neighbors including nodes B, C, ...,
$A \leftrightarrow B$	a symmetric link between node A and node B
$A \rightarrow B$	an asymmetric link from node A to node B
X	an attacker
$h(a)$	the Hash value of a
$i\|j$	the catenation of i and j
N	number of nodes in the network
T	an offline key server
U	upper bound of the number of nodes in the network
T_Max	upper bound of the lifetime of the network
CA	a CA server
$state$	a link state
l	the length (in bits) of Hash value
HC_k	the kth Hash chain
$seed_k$	seed of the Hash chain HC_k
h^m	the mth element in a Hash chain
$\mathfrak{L}$	the length of a Hash chain
$h^j(a)$	a value a hashed j times without key
$HC_{k;m}$	the mth Hash value of the Hash chain HC_k, it equals to $h^m(s_k)$
$from$	an advertising node
to	an advertised neighbor node
$interval$	the time interval of the creation of a "Link Atomic Information"
$n_1, n_2, ..., n_N$	the nodes in an N-node network
$cert_A$	the certificate of node A
K	a secret key
$T0$	the network starting time
$\{M\}_{key}$	the message M encrypted by the key key
t	a time
$interface_{n_j}$	an interface address of node n_j
IP_{n_j}	the main address of node n_j
H_i	the ith HELLO interval
$HELLO^A$	the previously received HELLO message from node A
$HC_{(A,B,state,i)}$	the Hash value which proves the existence of a link of type $state$ from A to B at time interval H_i
TC^A	the previously received TC message generated by node A
${}^xP_y$	number of permutations of x elements taken y at a time
${}^xC_y$	number of combinations of x elements taken y at a time

6.3 ADVSIG analysis

In this section, we analyze, in a more detailed way than in section 3.4.3.3, the security and the performance of the ADVSIG protocol.

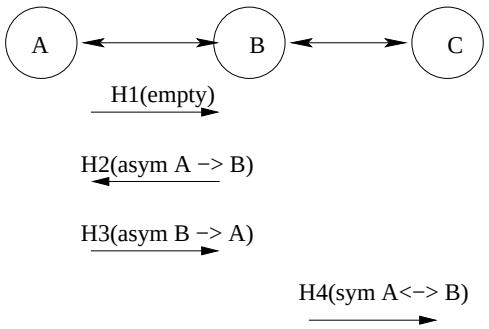


Figure 6.1: Example of ADVSIG

6.3.1 ADVSIG security analysis

ADVSIG is designed to guarantee the authenticity of network topology discovery against the attacks committed by intruders and compromised nodes, under the condition that there is no cooperating compromised nodes. It especially aims at countering the link spoofing attacks committed by compromised nodes. However, in the next subsection, our analysis shows that there could be a link spoofing attack even without colluding compromised nodes.

6.3.1.1 A security flaw

In ADVSIG, if a node A wants to declare in a HELLO message a link of type λ with node B, it should include in the message a *certificate* which is signed and provided by B in the previous HELLO message that is broadcasted by B. This *certificate*, also referred as a *proof* in the HELLO message of A, is composed of the address of A, the previous link state type λ^p of the link between A and B, a timestamp at the creation of the *proof*, and a signature of B. In ADVSIG, the different possibilities for the couple of (λ, λ^p) are the following:

- For λ = ASYM_LINK, no *proof* is required.
- For λ = SYM_LINK, λ^p = ASYM_LINK or SYM_LINK.
- For λ = SYM_NEIGH or MPR_NEIGH, λ^p = SYM_LINK or SYM_NEIGH.

In the following, we analyze the example presented in paper [RACM04] (note that this example is slightly different from the one presented in [Raf05], but our analysis holds for both examples). Figure 6.1 illustrates the example.
Let ϕ indicate that there is no *proof* or *certificate* possible, $< M >_{SK_B}$ be a message M signed by the private key of node B, $T_A(t_i)$ be the timestamp t_i of node A, and $<$ "A : *state*", $T_B(t_i) >_{SK_B}$ be a *proof* or a *certificate* signed by node B showing that at local time t_i of node B, B has a link of type *state* with node A. Let also an entire HELLO message format in ADVSIG be {certificate (link state) with the signature, *proof* with the signature, timestamp, signature}. We list the four ADVSIG messages that establish a symmetric link between A and B as follows[2]:

[2] All the examples that we show in this chapter only show the information that is necessary for the explaination of our examples. We ignore the other possible routing information in the messages.

1. $A \rightarrow \mathcal{N}(B) :< \phi,\ \phi,\ T_A(t_0) >_{SK_A}$
2. $B \rightarrow \mathcal{N}(A) :<< \text{“}A : ASYM\_LINK\text{”},\ T_B(t_1) >_{SK_B},\ \phi,\ T_B(t_1) >_{SK_B}$
3. $A \rightarrow \mathcal{N}(B) :<< \text{“}B : ASYM\_LINK\text{”},\ T_A(t_2) >_{SK_A},\ \phi,\ T_A(t_2) >_{SK_A}$
4. $B \rightarrow \mathcal{N}(C) :<< \text{“}A : SYM\_LINK\text{”},\ T_B(t_3) >_{SK_B},$
   $< \text{“}B : ASYM\_LINK\text{”},\ T_A(t_2) >_{SK_A},\ T_B(t_3) >_{SK_B}$

The example can be explained in detail in the following:

- At $T_A(t_0)$, node A broadcasts the HELLO message $H1$ that will be received by node B.
- B, in its next HELLO message $H2$, indicates with the status “ASYM_LINK” that it can hear A. Upon receiving $H2$, A obtains the signature of node B which attests that at time $T_B(t_1)$, there exists an asymmetric link from A to B.
- The HELLO message $H3$ is similar to $H2$, which attests an asymmetric link from B to A. A symmetric link is then established between A and B, and both two nodes have a signature signed by the other.
- The message $M4$ that B sends to C (a neighbor of B) confirms the symmetric link $A \leftrightarrow B$ to C, thanks to the *proof* signed by node A. Therefore, C can be sure that A is its symmetric 2-hop neighbor, if it has no direct link with A.

If the three nodes in figure 6.1 are independent, no security flaw exists in the schema, because A knows the existence of B only when it can receive $H2$ from B. However, we note that HELLO messages are not unicasted but broadcasted. Therefore, in the case of the topology shown in figure 6.2, X can fool B and C and make them believe in the existence of a symmetric link $X \leftrightarrow B$ while there is only an asymmetric link $X \rightarrow B$.
The attack can be described as follows. Attacker X starts by sending $H1$ to B, B will then try to reply to X with $H2$. However, if there is only an asymmetric link from X to B, $H2$ will not be received by X but will be received by node D. Later, D will reply with a HELLO message $H3$ to B containing some information about B that will also be heard by X (suppose that there is at least an asymmetric link from D to X). Therefore, even if X cannot hear directly from B, X knows the existence of B in no more than 2 hops away. Then X can try to send $H4$ that B will receive due to the link $X \rightarrow B$. Afterwards, $H5$ can be sent to C. As a consequence, both C and B will believe in the existence of a symmetric link $X \leftrightarrow B$.
Let $X \rightarrow \mathcal{N}(B) : M$ indicate that X broadcasts a message M to its neighbors including B, the scheme of the attack can be shown as follows:

1. $X \rightarrow \mathcal{N}(B) :< \phi,\ \phi,\ T_X(t_0) >_{SK_X}$
2. $B \rightarrow \mathcal{N}(D) :<< \text{“}X : ASYM\_LINK\text{”},\ T_B(t_1) >_{SK_B},\ \phi,\ T_B(t_1) >_{SK_B}$, note that this message cannot be received by X

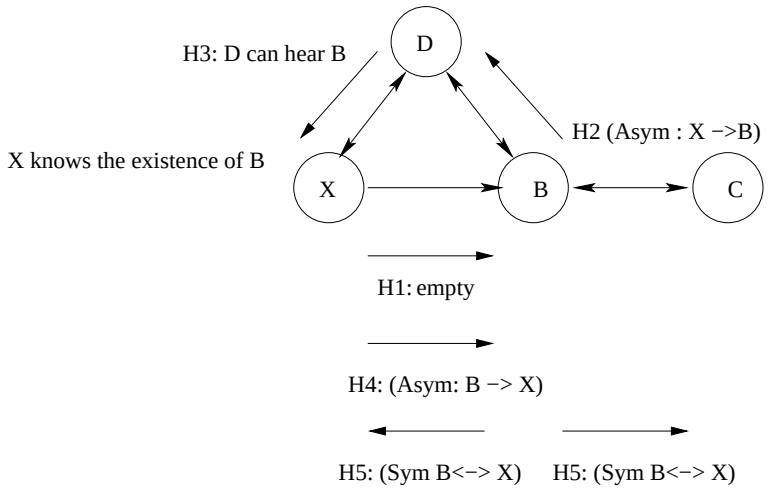


Figure 6.2: Example illustrating the flaw in ADVSIG

3. $D \rightarrow \mathcal{N}(X) :<<$ "$B : ASYM\_LINK''$, $T_D(t_2) >_{SK_D}$, ϕ, $T_D(t_2) >_{SK_D}$

4. $X \rightarrow \mathcal{N}(B) :<<$ "$B : ASYM\_LINK''$, $T_X(t_2) >_{SK_X}$, ϕ, $T_X(t_2) >_{SK_X}$

5. $B \rightarrow \mathcal{N}(C) :<<$ "$X : SYM\_LINK''$, $T_B(t_3) >_{SK_B}$,
   $<$ "$B : ASYM\_LINK''$, $T_X(t_2) >_{SK_X}$, $T_B(t_3) >_{SK_B}$

6.3.1.2 Attack analysis

The attack described in section 6.3.1.1 can take place because no *proof* verification is required in the declaration of an asymmetric link. Thus X can forge the message $H4$ to make B believe that it can hear B. Then, only the link direction from X to B is really verified[3].
Due to the attack, B and its neighbors will believe in the existence of the link $X \leftrightarrow B$ during several seconds (according to some experiences, that will be about 30 seconds). Then B will possibly be chosen as MPR by its neighbors to reach X, and reversely X may also be chosen as MPR by B (however due to the asymmetric link, X will not be able to send a TC message which reports B as a MPR selector). As a result, there could be data losses, and the topology of the network is not seen correctly by nodes, neither.
However, since after the attack there would be HELLO messages sent by B that X cannot correctly reply, the attack has only temporary consequence and the forged link will become a "LOST_LINK" in a few seconds.

6.3.1.3 ADVSIG improvement: ADVSIG$^+$

In order to counter the above security flaw, we suggest an improved ADVSIG called ADVSIG$^+$. In ADVSIG$^+$, the declaration of an asymmetric link also requires a *proof*.

[3]In the example in [Raf05], the same problem exists since in the second message no *proof* can tell that B can really hear from X.

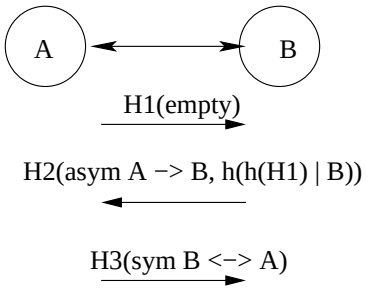


Figure 6.3: Example of ADVSIG$^+$

In figure 6.3 we illustrate the standard neighbor establishment dialog in ADVSIG$^+$:

1. $A \rightarrow \mathcal{N}(B) :< \phi,\ \phi,\ T_A(t_0) >_{SK_A}$, A saves $h(H1)$
2. $B \rightarrow \mathcal{N}(A) :<< \text{“}A : ASYM\_LINK''$, $T_B(t_1) >_{SK_B}, h(h(H1)|B)$, $T_B(t_1) >_{SK_B}$, A verifies $h(h(H1)|B)$
3. $A \rightarrow \mathcal{N}(B) :<< \text{“}B : SYM\_LINK''$, $T_A(t_2) >_{SK_A}$, $< \text{“}A : ASYM\_LINK''$, $T_B(t_1) >_{SK_B}$, $T_A(t_2) >_{SK_A}$

The example can be explained as follows.

- Upon receiving the HELLO message $H1$, B is sure that it can directly hear from A (we do not consider the replay attacks). Then it declares an asymmetric link $A \rightarrow B$, in which it includes the Hash value of the combination of $H1$ and the identity of B, $h(h(H1)|B)$, as a proof.

  Thanks to the utilization of $h(H1)$, nodes only need to store a Hash value instead of a whole HELLO message. Note also that the identity of B is needed in the hash, otherwise the proof might be reused by another node to declare an asymmetric link with B. Even though $h(h(H1)|B)$ cannot be verified by the two-hop neighbors of B, the operation of ADVSIG$^+$ is not influenced since asymmetric links are not considered by two-hop neighbors.

- Upon receiving the message $H2$, A can be sure that B can really hear it if A can successfully verify $h(h(H1)|B)$. Other neighbor nodes of B receiving $H2$ are not able to verify the proof since they do not know $h(H1)$.

- The third message is the same as in ADVSIG.

With ADVSIG$^+$ the attack presented in section 6.3.1.1 is no more possible. This is because, even though X knows the existence of B, it cannot forge the *proof* $h(h(H1)|B)$ that is required to declare the asymmetric link $X \rightarrow B$, since it cannot receive $H1$ from B. X can neither recompute a valid *proof* by receiving messages from D, unless D colludes with it.

6.3.2 ADVSIG computational overhead analysis

In ADVSIG (also in ADVSIG$^+$), cipher choice should be carefully made, since every node should sign and verify a large number of asymmetric signatures in each HELLO interval (2 seconds as default setting) and in each TC interval (5 seconds as default setting).

According to the authors of OLSR and our simulations, an HELLO or a TC message advertises on average 9 neighbors. Thus, in ADVSIG about 10 signatures (9 *certificates* and one global signature) should be computed for generating each HELLO message, and one signature should be computed for generating each TC message.

In addition, a node will receive on average 9 HELLO messages in one HELLO interval, among which on average 5 messages contain a *proof* that needs to be verified. Thus each node has to do about $(5+1)\times 9 = 54$ signature verifications in every HELLO interval.

For an OLSR network that contains N nodes, a node receives in addition a maximum of $N-1$ TC messages in each TC interval. Therefore a maximum of $(N-1)\times(9+1) \sim 10N$ signature verifications are to be performed in every TC interval by each node.

Since mobile ad hoc nodes are often heterogeneous and range from laptops, handsets, PDAs to sensors, some of them may fail in affording heavy cryptographic operations. Thus, in the networks where the processing power is limited, ADVSIG (and ADVSIG$^+$) is expensive or even prohibited in terms of traffic, processing overhead and energy cost due to the important number of asymmetric cryptographic operations required. Therefore, our main motivation is to reduce the computational overhead of ADVSIG.

6.4 Our approaches to secure OLSR

In this section, we introduce two approaches to secure OLSR. They can prevent both external and internal attackers from injecting incorrect routing information into network, and they are much more lightweight than ADVSIG in terms of computational overhead and control message overhead. However, their main idea is similar to ADVSIG: in order to make a link/MPR information be validated, it has to be confirmed by the two ends of the link.

Our first approach is an add-on security mechanism which can be applied to both HELLO and TC messages. The second approach slightly modifies the basic OLSR and is only applicable to TC messages, but it can be combined with the other mechanisms to form a complete OLSR security solution.

6.4.1 Assumptions

In this subsection we introduce the common assumptions of the two approaches. Their specific assumptions will be introduced later.

6.4.1.1 Network assumptions

As in OLSR, we do not assume that all links are bi-directional, since in ad hoc networks uni-directional links can exist due to many factors, such as difference in radio emission power, directional antenna, obstacle, etc [Per01, MM04]. In OLSR, unidirectional links are not included in routing tables but they are only used in the establishment of symmetric links.

As in ADVSIG, we suppose that each node is able to provide correct timestamps thanks to a synchronization within the network (in other words, each node should have a same or nearly the same local time clock). It is out of the scope of this thesis to study the synchronization problem, but a lot of synchronization methods have already been proposed for MANETs in the literature [LZ03, SV04, Soc05]. Thus, we suppose that one of them could be used and be secured against attacks.

If ever a network-wide synchronization is not available, each node should save the last timestamps of the other nodes and be able to estimate the local time of the other nodes. This assumption allows nodes to immediately judge the freshness of the messages sent by the other nodes even under a time shifting.

6.4.1.2 Node assumptions

As a first step, we assume that all nodes wishing to communicate with others will fully participate in the ad hoc routing[4]. In other words, we assume that all nodes in the network will regularly send their HELLO messages and, if necessary, also their TC messages[5,6].

We assume the existence of compromised nodes, and we suppose that they can declare forged routing information in their HELLO and TC messages. But we do not consider colluding compromised attackers.

To make our security mechanisms as adaptive as possible, we do consider nodes with minimal resources in our design. We suppose that the resources of different ad hoc nodes can vary largely.

[4]Here we do not consider the selfish nodes which show their willingness of not being MPR in their HELLO messages.

[5]Indeed, in OLSR a node refusing to send HELLO message cannot establish asymmetric nor symmetric links with its neighbors. And, if HELLO messages are not sent regularly, even established links can be lost after a holding time (a timeout). Therefore such a node will have difficulty in entering the routing tables of the other nodes, and it might not be reached by traffics as an intermediate node or a destination node.

However, a node neglecting TC messages can avoid being MPR thus not being on routes as an intermediate node. The network will also be less connected, and the nodes that have chosen selfish nodes as their only MPR nodes will be isolated.

We do not consider the selfish behaviors in our propositions, because the selfishness usually cannot be countered by cryptographic measures. We suppose that a node not refusing to be MPR will always correctly send TC messages following the standard OLSR.

[6]We will present a mechanism in section 6.4.3 that can naturally prevent nodes from not sending their TC messages.

6.4.1.3 Security assumptions

As in ADVSIG, the existence of a PKI is assumed (a PKI can be established in ad hoc networks thanks to the schemes described in section 3.3.2). Each node has at least a pair of asymmetric keys with which it can sign messages. Moreover, all public keys are known to all nodes[7], therefore all signed messages can be verified by all nodes in the network, for their authentication, non-repudiation and integrity.

6.4.2 First approach: Hash Proved Link State (HPLS)

In this subsection, we propose an approach named *Hash Proved Link State* (HPLS), which requires an offline server. In HPLS we use Hash values to replace the multiple digital signatures in ADVSIG.

6.4.2.1 Additional assumptions

We assume the existence of an offline server T which should have a necessary computational and storage capacity.
T should be able to estimate in advance and at least in rough figures the upper bound U ($U \geq 2$) of the number of nodes in the network. We assume that the wireless local network has a reasonable size, thus U will not be too large.
T should equally be able to estimate the upper bound of the lifetime of network T_Max. T_Max is then divided into a number of uniform time intervals, which duration is equal to that of OLSR HELLO intervals. Note that this assumption can be removed if the server T can be online to redistribute cryptographic credentials.
We assume that either T knows the identities of the nodes, or alternatively, it plays the role of a Dynamic Host Configuration Protocol (DHCP) server [Dro97], or it can cooperate with a DHCP server to achieve the maintenance of node identities.
Furthermore, to simplify the scheme, we may even suppose that four servers, respectively a synchronization server, a CA (for issuing/renewing certificates) server, a DHCP server, and T are all installed together and can securely communicate among them.
We equally suppose that the offline server T can securely communicate with the nodes. For instance, T may have a pair of asymmetric keys. Otherwise, nodes and T may use an infrared or physical contact module, or exchange some memory devices such as smartcard, memory card, USB key, etc, to ensure the security of the communication.
Finally, we assume that nodes are able to do HMAC operations, and the HMAC algorithm used by them is collision resistant (c.f. section 4.5).
In OLSR, multiple-interface nodes are considered. However, in HPLS, as a first step we only consider single-interface nodes. The problems caused by multiple-interface nodes will be discussed in section 6.4.2.4.

[7] In [Raf05], a proactive PKI is proposed for ADVSIG, wherein an authority periodically broadcasts the public keys of the nodes in the network.

6.4.2.2 Basic idea

In ADVSIG (c.f. sections 3.4.3.3 and 6.3), a *proof* should systematically contain four elements: the (interface) address of the originator node, the (interface) address of the advertised node, the link state to the originator('s interface) with respect to the advertised node('s interface), and the creation time of the *proof*. Additionally, to provide the guarantee of integrity and authentication, a signature on the *proof* should be computed by the advertised node. The four elements together with the signature are called "Link Atomic Information" in ADVSIG.
Our basic idea is to replace the signature in the "Link Atomic Information" by a Hash value, in order to achieve the following improvements:

- The security mechanism has **lower computational overhead** since a Hash value can be more easily calculated and verified.
- The mechanism has **lower routing overhead**, since the length of Hash value can usually be much shorter than an asymmetric signature.

We require that a Hash value in HPLS should

- be used to **represent and confirm the four elements** in the "Link Atomic Information". This will allow nodes to fight against link forging attacks committed by external and internal attackers, thus ensuring the authenticity of the link information.
- be **unique** for each link state between each couple of nodes at each HELLO_INTERVAL. Thus, it cannot be reused by attackers.
- be **secret** before its corresponding time interval. In other words, before its disclosure, it is only known by its provider. Therefore the authentication of the link information can be guaranteed with the Hash value.

6.4.2.3 Scheme

In addition to the notations defined in section 6.2, we further define the five states that *state* can represent as follows:

$$state = \begin{cases} 0, & \text{NO\_LINK} \\ 1, & \text{LOST\_LINK} \\ 2, & \text{ASYM\_LINK} \\ 3, & \text{SYM\_LINK and SYM\_NEIGH} \\ 4, & \text{SYM\_LINK and MPR\_NEIGH} \end{cases}$$

HPLS HELLO/TC message format

The HPLS HELLO and TC message formats are shown in figures 6.4 and 6.5, the fields different to the original OLSR protocol are noted in italic.

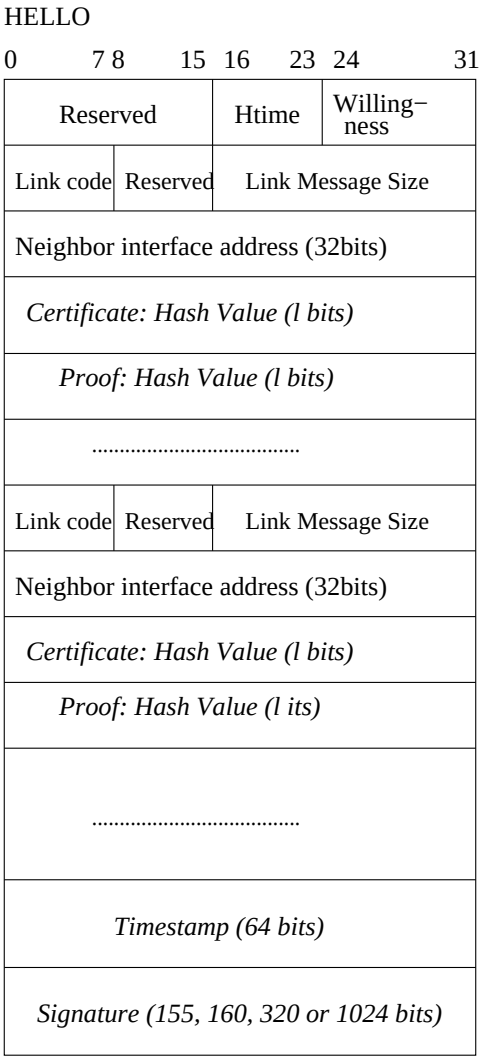


Figure 6.4: HPLS HELLO message format

Compared to the classical HELLO message format, to each link information we add two l-bit Hash values, one is used as a *certificate* and the other one is used as a *proof*. According to the network size and the required security level, l can be set to 64, 96, 128, 160 or even larger. The larger is the network size, the higher is the required security level, and the larger should l be.
In addition, we add two global fields to each HELLO message. They are respectively a 64-bit timestamp and a digital signature. The length of the signature depends on the signature algorithm in use[8].
Compared to the classical TC message format, one l-bit Hash value is added to each MPR selector address as a *proof*. Moreover, two global fields, respectively a 64-bit timestamp and a signature, are added to each TC message.
Thus far, our modifications to OLSR control messages are similar to ADVSIG. How-

[8]For example, for DSA 1024 it will be 320 bits, for $ECNR\ GF(p)$ 155 it will be 336 bits, for $ECNR\ GF(2^n)$ 168 it will be 310 bits, and for RSA 1024 it will be 1024 bits.

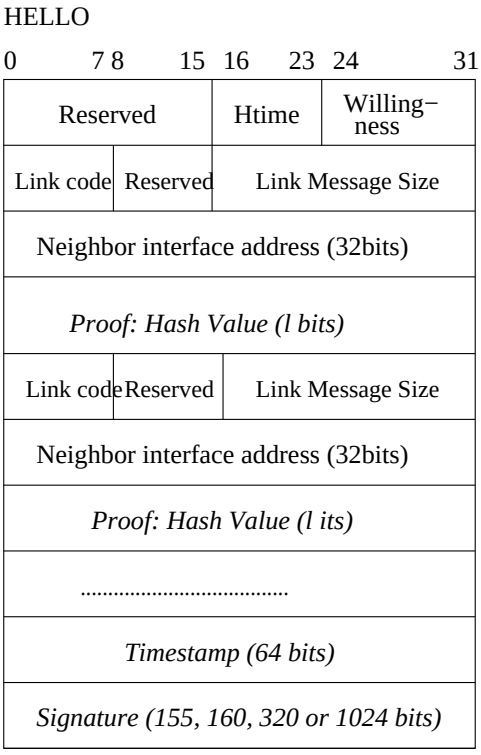


Figure 6.5: HPLS TC message format

ever, different to ADVSIG, we do not add a *timestamp* field for each *proof*. This is because we use time intervals instead of timestamp, and the possible time intervals are very limited in HPLS – it has only NEIGHB_HOLD_TIME/HELLO_INTERVAL (3 by default) possibilities. Then, at the price of some (from 0 to 2) additional hash operations, we can check the validity of *proofs* without the information of time interval (we can test the possibilities one-by-one until one time interval is validated). This can greatly reduce the length of message headers, and does not introduce new vulnerabilities, since

- There is no relationship between the different Hash chains. For example, a malicious node cannot obtain a *proof* for SYM_LINK if it only has a *proof* for ASYM_LINK, and vice versa.
- There are only one-way hash relationships between the values in a same Hash chain. For example, if the Hash value of type SYM_LINK is declared in a time interval, attackers cannot know the Hash value of type SYM_LINK in the next time interval.
- The Hash values cannot be replayed even though indicated the corresponding hash interval is not indicated. Any Hash value older than TOP_HOLD_TIME will be considered as expired.

Moreover, different to ADVSIG which creates an additional message for each HELLO and TC message, we plan to add link and security information into one alone

HELLO or TC message. Only when the message size limitation is surpassed, a new HELLO or TC message is created, and naturally all will be sent within one REFRESH_INTERVAL (the default value of REFRESH_INTERVAL is 2 seconds, same as HELLO_INTERVAL). We believe that this difference with ADVSIG can bring us the following advantages:

- The overall routing overhead can be less important. We found during our simulations that message size has less performance impact on the packet delivery ratio than message quantity. Thus, instead of increasing the number of messages, we choose to increase the size of messages.

- The packet loss can have less influence on the authenticity of topology than in ADVSIG, where the loss of any of the routing control message and the corresponding ADVSIG message makes the other message useless.

- Each link/neighbor information can be verified immediately, without waiting for the arrival of the corresponding ADVSIG message.

However, if it is important to keep the original format of OLSR, the additional fields of HPLS can also be treated as in ADVSIG. This is to say, they can be sent separately in a different message.

Server initiation

To represent three (ASYM_LINK, SYM_LINK and SYM_NEIGH, SYM_LINK and MPR_NEIGH) of the five link/neighbor states between each couple of nodes, the server calculates $U \times [3 \times (U-1)] = 3U^2 - 3U$ Hash chains denoted as HC_1, ..., HC_{3U^2-3U} using $3U^2 - 3U$ different seeds denoted as $seed_1$, $seed_2$, $seed_3$, ..., $seed_{3U^2-3U}$. Each Hash chain HC_k contains $\mathfrak{L} + 1$ elements: $seed_k$, $HC_{k;1}$, ..., $HC_{k;\mathfrak{L}}$. In figure 6.6, we show a server initiation example, wherein for a MANET which has at maximum three nodes ($U = 3$), the server computes 18 Hash chains during its initiation phase. Note that we do not create Hash chains for "NO_LINK" and "LOST_LINK" since they do not need to be proved.
In HPLS, the "Link Atomic Information" is slightly different to the one in ADVSIG, since we do not consider multiple interface nodes, and we use time interval information instead of timestamp. Our "Link Atomic Information" is composed of the four following elements: ($from$, to, $state$, $interval$), where $from$ is the advertising node, to is the advertised neighbor node ($from \neq to$), $interval$ is the time interval of the creation of the "Link Atomic Information", and $state$ is the link state from $from$ to to at the time interval $interval$.
As shown in figure 6.6, each Hash value is able to uniquely represent a set of four elements of a "Link Atomic Information", since the position of any Hash value maps bijectively to a "Link Atomic Information".

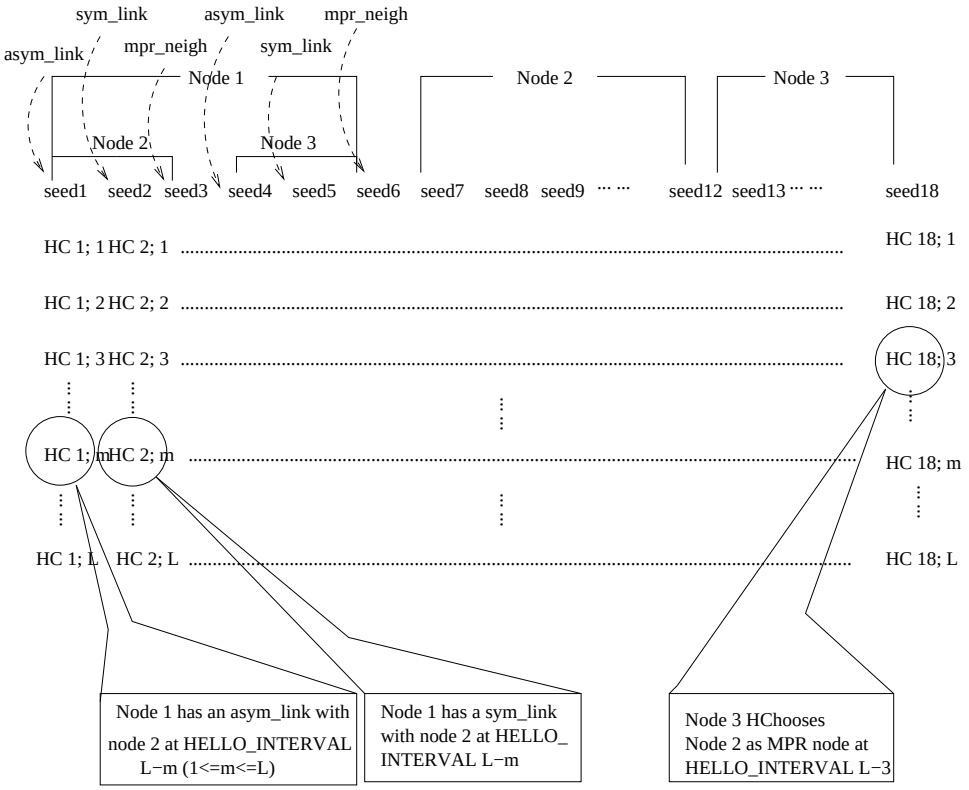


Figure 6.6: Hash chains generated by HPLS server during the initialization: $U = 3$

Node bootstrap

Before entering in the network, each node needs to contact the server, and the sever will map it to a set of Hash chains and secretly communicate the seeds of these chains to it.
The node initiation can be realized with a two-way handshake. For this, we propose two messages, a Key REQuest (KREQ) message from a mobile node to the server, and a Key REPly (KREP) message from the server to a mobile node.
Let K be a random secret generated by node n_j, each node n_j starts the process by sending a KREQ message to server T:

$$n_j \rightarrow T :< \ cert_{n_j}, \ \{K\}_{PK_T} >_{SK_{n_j}}$$

K is encrypted by the public key PK_T of the server. The whole KREQ message is signed by node n_j.
Upon receiving the KREQ, after the verifications of the certificate $cert_{n_j}$ and of the message signature, T replies to n_j with the following message (let t be the current time and $T0$ be the network start time):

$$T \rightarrow n_j :< \ U, \ \{seed_{(j-1)(3U-3)+1}\}_K, ..., \ \{seed_{j(3U-3)}\}_K, \ T0, \ t >_{SK_T}$$

For confidentiality reasons, the seeds transported in the message are encrypted with the secret key K. Therefore within the KREP, the node n_j will securely receive

$3U-3$ seeds and use them to compute $3U-3$ Hash chains.
The server should also communicate the last elements of all the Hash chains $HC_{1;\mathfrak{L}}$, ..., $HC_{3U^2-3U;\mathfrak{L}}$ to all the nodes which enter the network. Indeed, the last elements can be broadcasted to the network[9]:

$$T \rightarrow * :< HC_{1;\mathfrak{L}}, ..., \ HC_{3U^2-3U;\mathfrak{L}} >_{SK_T}$$

These values can mainly be used by each node to verify the *proofs* issued by the other nodes. In addition, they can also be used to verify the correctness of the computations of the Hash chains belonging to each node.

If (as mentioned in section 6.4.2.1) multiple servers are installed on T, or multiple servers can cooperate and T is the interface between the nodes and the servers, then to achieve the management of identities and the distribution of cryptographic credentials at the same time, the KREQ message could be somewhat similar to the following one (let K be a secret key, and PK_{n_j} and SK_{n_j} be the asymmetric keys of node n_j):

$$n_j \rightarrow T :< \quad PK_{n_j}, \ \{K\}_{PK_T} >_{SK_{n_j}}$$

Then T should reply to n_j with the following message:

$$T \rightarrow n_j : \{ \ U, \ n_j, \ cert_{n_j}, \ \{seed_{(j-1)(3U-3)+1}\}_K, ..., \ \{seed_{j(3U-3)}\}_K, \ T0, \ t\}_{SK_S}$$

All the certificates should then be published in the network (suppose that the network is composed of N nodes):

$$T \rightarrow * :< cert_1, ..., \ cert_N >_{SK_T}$$

Hash value and Certiproof tables

In HPLS, each node n_j should maintain some tables to store the information received from the server and the other nodes:

- Let k $((j-1)(3U-3)+1 \leq k \leq j(3U-3))$ be the number of a Hash chain, and $HC_{k;\mathfrak{L}-i}$ be the current Hash value (in the current time interval $\mathfrak{L}-i$) of chain HC_k. For the chains belonging to n_j, they are stored by n_j in a *Local Hash Value Table* in the following form:

  $$< k, \ seed_k, \ HC_{k;\mathfrak{L}-i} >$$

  Otherwise, the Hash chains can also be computed and stored as in [Jak02], where we try to find a compromise between the storage requirement and the computational overhead of the utilization of Hash chains.

[9] We may use the proactive PKI that is proposed by ADVSIG to realize it.

- Let k $(1 \leq k \leq (j-1)(3U-3)$ or $j(3U-3)+1 \leq k \leq 3U^2-3U)$ be the number of a Hash chain, and $HC_{k;\mathfrak{L}}$ be the last element of the chain HC_k that is published by the server. To save information regarding the other Hash chains, a *Foreign Hash Value Table* can be maintained by n_j with the tuples as follows:

  $$< k,\ \text{last revealed value},\ \text{index of the last revealed value},\ HC_{k;\mathfrak{L}} >$$

  The *last revealed value* is the most recently revealed Hash value of chain HC_k that is heard by node n_j, and *index of the last revealed value* is its index in the chain.

- To store the *certificates* sent by the other nodes, node n_j maintains a *Certiproof Table*, which has the same format as in ADVSIG, except the fields *signature* and *timestamp*:

  $$< \text{originator},\ \text{advertised node},\ \text{link state},\ \text{interval},\ \text{Hash value} >$$

  The field *originator* is the key of the table. For each *originator*, only the newest tuple is kept. Furthermore, any tuple expires after three time intervals, which corresponds to the TOP_HOLD_TIME in OLSR.

- Finally, node n_j should keep a table of mapping between the identities of nodes and the number of nodes:

  $$< \text{node number}, \text{node identity} >$$

HELLO/TC message generation

In our first approach, we use the principle of ADVSIG$^+$ to secure HELLO and TC messages. Moreover, we replace the *certificates* and *proofs* in ADVSIG$^+$ by the corresponding Hash chain elements.
In addition to the standard operations on the original HELLO/TC message fields, to generate a HELLO or a TC message at time interval H_i, a node $from$ should:

1. Write the current time t into the *Global Timestamp* field.
2. If it generates a HELLO message, for each *Neighbor Interface Address* $interface_{to}$ with the link state *state*,
   (a) Find the corresponding main address IP_{to} of $interface_{to}$.
   (b) If it is necessary (*state* is one of ASYM_LINK, SYM_LINK and SYM_NEIGH, SYM_LINK and MPR_NEIGH), find and copy the corresponding Hash value (c.f. section 6.4.2.3) into the *certificate* field following the link information of $interface_{to}$.
   (c) Update its *Local Hash Value Table* with the copied Hash value.

(d) If *state* equals ASYM_LINK, $from$ computes $h(h(HELLO^{to})|from)$ as a *proof*, where $h(HELLO^{to})$ is the hash of the previous HELLO message received from *to*. If *state* equals "SYM_LINK and SYM_NEIGH" or "SYM_LINK and MPR_NEIGH", $from$ finds the corresponding *proof* in the *Certiproof table*. Then, with the found value $from$ fills the *proof* field following the link information of $interface_{to}$. Note that since only the last *certificate* from each node is kept in the *Certiproof table*, the *proof* is unique.

3. If it generates a TC message, it finds the corresponding hash value (always with *state* equals MPR_NEIGH) in the *Certiproof table* and copies it into the *proof* field following the link information of $interface_{to}$.

4. Compute the *Global Signature* on the whole message.

5. Save the hash of the message.

6. Send out the message.

HPLS does not create Hash chains for every interface address. Instead, it always uses Hash values according to the main addresses of nodes. We will show in section 6.4.2.4 that this choice will not threaten the security of OLSR.

HELLO/TC message processing

When a node *to* having the main address IP_{to} receives a HELLO/TC message from an originator $from$ at interval H_i, it executes the following algorithm:

1. Check the validity of the *Global Timestamp* field.

2. Check the validity of *Global Signature* by using the public key of $from$.

3. If it is a TC message, for each *Advertised Neighbor Main Address*, *to* will (note that with TC messages we only use the main addresses of nodes):

   (a) With some hash operations, check the validity of the hash value used as *proof*.

   (b) If the previous step is successful, update the *Foreign Hash Value Table* with the Hash value.

4. If it is a HELLO message, *to* will

   (a) Save $h(HELLO)$ (note that the expiration time of this storage is NEIGHB_HOLD_TIME).

   (b) For each *Neighbor Interface Address*, $interface_{to}$, with link state *state*, *to* will:

      i. Find the corresponding *Neighbor Main Address to*.

ii. If *state* is ASYM_LINK, check the validity of $h(h(HELLO^{to})|IP_{from})$, where $HELLO^{to}$ is the previous HELLO message sent by *to*. If *state* is one of SYM_NEIGH and MPR_NEIGH, check with some hash operations the validity of the Hash value used as *proof*, where the previous state λ_p should answer to the requirements described in section 6.3.1.1. Update the *Foreign Hash Value Table* at the same time.

iii. If the above operations are successful, update the *Certiproof Table* with the Hash value used as *certificate*.

Standard dialog

Suppose that the first message is sent in time interval H_i, then the standard dialog which establishes a symmetric link between node A and B in HPLS can be shown as follows (let $HC_{(A,B,state,i)}$ be the Hash value which proves the existence of a link of type *state* from A to B at time interval H_i):

1. $A \rightarrow \mathcal{N}(B) :< \phi,\ \phi,\ T_A(t_0) >_{SK_A}$
2. $B \rightarrow \mathcal{N}(A) :<< \text{“}A : ASYM\_LINK'' >, HC_{(B,A,2,i)}, h(h(HELLO^A)|B),\ T_B(t_1) >_{SK_B}$
3. $A \rightarrow \mathcal{N}(B) :<< \text{“}B : SYM\_LINK'' >, HC_{(A,B,3,i+1\ or\ i)}, HC_{(B,A,2,i)},\ T_A(t_2) >_{SK_A}$

Since a Hash operation is 10^3 to 10^4 times faster than an asymmetric signature or a signature verification operation, our solution can significantly reduce the computational overhead of ADVSIG.
However, instead each node should either do a large number of hash operations or store $3 \times (U-1)$ Hash chains. We argue that the storage and computational requirements can be balanced with the mechanism proposed in [Jak02], where each node only needs to store $O(U \times 3 \times log_2(\mathfrak{L}))$ Hash elements – that means in most cases 10KB - 30KB memory can meet the requirement.

6.4.2.4 Security analysis

In this section, we show that HPLS can achieve the same security level as ADVSIG$^+$.
In HPLS, each routing message carries a *Global Timestamp* and a *Global Signature* that will be used to verify the authentication, the integrity and the freshness of the message. In addition, a *proof* is appended to each *Advertised Neighbor Address* which authenticates the main address of the sender, the link state between the sender and the advertised node, and the time interval of the creation of the *proof*. Since a *proof* can only be issued by the *Advertised Neighbor Node*, each link/neighbor/MPR relationship is then confirmed by its two ends.
In addition, we also use the hash of the previous HELLO message sent by the *Advertised Neighbor Node* to confirm an asymmetric link. As a result, HPLS can have the same security level as ADVSIG$^+$.

Interface address vs. Main address

Computing asymmetric signatures on "Link Atomic Information" permits ADVSIG to takeinterface addresses of nodes into consideration. From this point of view, our *certificates* (Hash values) are less dynamic since they can only be computed on the corresponding *Neighbor Main Addresses*.
In a network where all nodes are single-interface nodes, the problem does not exist. Only when the network contains multiple-interface nodes, and a *Neighbor Interface Address* happens not to be the main address of a neighbor node, the meanings of the two *proofs* resulted by the two approaches are different. For example, suppose that at time t a node having as main address IP_A sends a HELLO message to another node B describing a link state *state* with one of B's interfaces $interface_B$, the *certificate* in ADVSIG will be computed based on $(A, C, state, t)$, while in our approach, the *certificate* will be based on $(A, B, state, t)$. Nevertheless, this fact will not decrease the security level of HPLS, because:

- In HPLS the *certificate* can only be used by B as a *proof*. Other nodes cannot use it since they have their main addresses different to B.
- If B is an attacker and it has another interface $interface'_B$ ($interface'_B \neq B \neq interface_B$), B may send via the interface $interface'_B$ a HELLO message containing A as a *Neighbor Address* and the above *certificate* as the corresponding *proof*, in order to create a forged link between $interface'_B$ and A.

  However, the attack cannot influence the security of OLSR, since according to the link set update algorithm of OLSR, if A cannot receive the message from the interface $interface'_B$, the link between $interface'_B$ and A will not be registered by A. Otherwise, if A can really receive the message from the interface $interface'_B$, then the link $A \leftrightarrow interface'_B$ exists[10].
- In OLSR any third node only cares whether a link between A and B exists, it does not care if the link is between A and $interface'_B$, A and B or A and $interface_B$.
- For TC messages, only main addresses are concerned thus the difference between interface addresses and main addresses is not important.

However, as in ADVSIG$^+$, the attacks committed by colluding compromised nodes are not countered in our approach.

6.4.2.5 Summary

We believe that our solution has the following advantages compared with ADVSIG:

- It uses Hash values instead of signatures. This can generate a great gain in terms of computational overhead and routing overhead.

[10] Here we do not consider the interface address spoofing attacks which exist also in ADVSIG, since there is only one asymmetric key pair per node.

- The original OLSR protocol is unmodified. HPLS can be an add-on mechanism to OLSR.

However, it also has the following disadvantages:

- It requires an offline server which should be able to estimate in advance the number of nodes in the network and the network lifetime. In addition, nodes should be able to securely communicate with the server during their bootstrap.
- The number of Hash chains increases rapidly with the number of nodes in the network.

6.4.3 Second approach: Securing TC messages (TCSec)

We propose a second approach that could be used to secure TC messages. It does not directly secure HELLO but it can be combined with other mechanisms such as ADVSIG or HPLS to provide a global security solution to the OLSR protocol. We call this approach Securing TC messages (TCSec).

6.4.3.1 Additional assumptions

In addition to the assumptions mentioned in section 6.4.1, we assume that for TCSec, HELLO messages are already secured by some means or other (either ADVSIG or HPLS can be applied only to HELLO messages but not to TC messages). Therefore only authentic one-hop and two-hop neighbor information and MPR information can be provided to nodes.

6.4.3.2 Basic idea

In the standard OLSR, a node does not generate TC if it is not chosen as MPR, it is only required to send at least one TC message per TOP_HOLD_TIME which is by default equal to three times of TC_INTERVAL.
However, to realize our second approach, we need to slightly modify the above requirement. We will add some additional routing fields to TC messages, and we also require that every node send one TC message per TC_INTERVAL.
Our new TC header is shown in figure 6.7 (the fields in italic are the new fields added by TCSec). It contains all fields of the classical TC header, and adds the *MPR Set* of the sender node, the size of the *MPR Set*, a timestamp and a signature to each TC message. The timestamp indicates the time of the generation of the TC message, and the signature is computed on the sequence of bits made up of the whole TC message.
The basic idea of TCSec is illustrated in figure 6.8. If a node A has chosen another node B as MPR, not only should B send a TC message including A in its MPR Selector Set, but also A should send a TC message including B in its MPR Set. As a result, after both TC messages are received by a third node C, C can be sure of the MPR relationship thanks to the confirmations of both ends. Any MPR relationship declared by only one node will be regarded as invalid and will not be used to calculate routing tables.

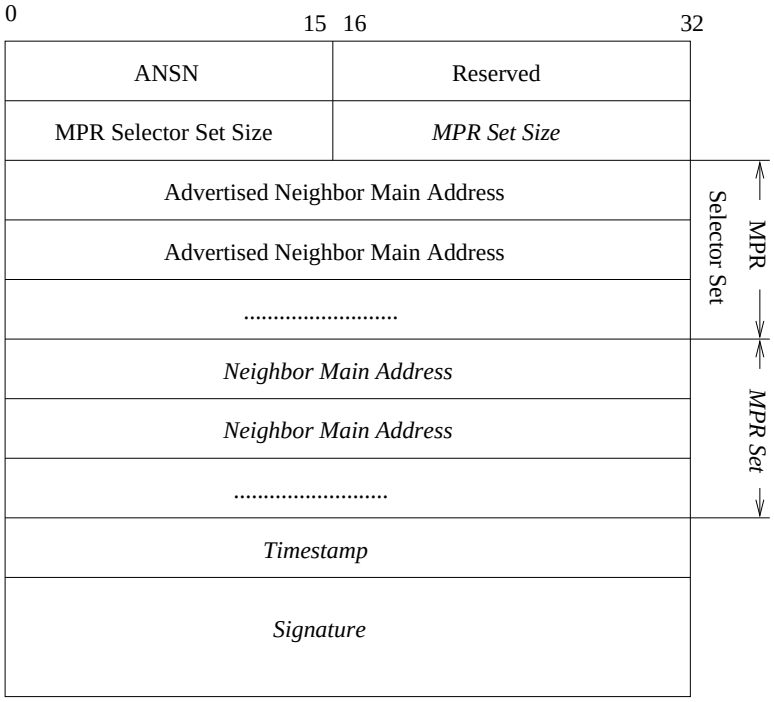


Figure 6.7: TC message format in TCSec

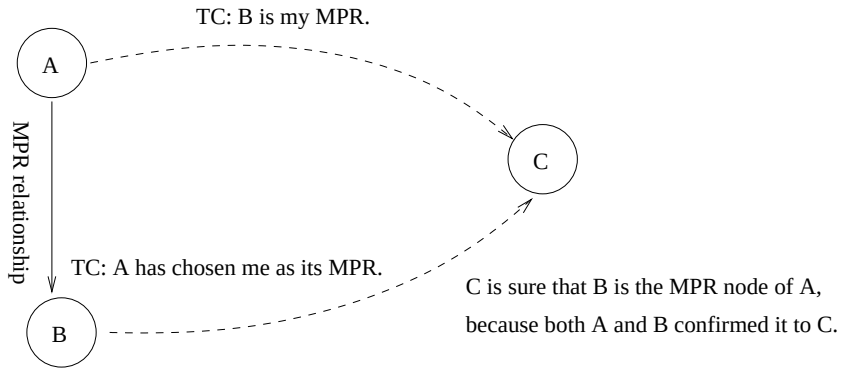


Figure 6.8: Basic idea of TCSec

6.4.3.3 Scheme

In this section we draw an outline of the TCSec scheme, and then we detail the TC message processing in TCSec.

TC message generation: Every node generates at least one TC message per TC interval. It writes its *MPR Selector Set* and its *MPR set* into the message, and sets the "MPR Selector Set Size" and the "MPR Set Size" to indicate the numbers of the addresses in the two sets. Later it puts the current time in the timestamp field and signs the whole message using its private key.

Last Received TC message Set: Each node in the network maintains a *Last Received TC Message Set* consisting of the last TC message(s) received within the last TC_INTERVAL from every other node. Each TC message in the set is stored in form of *Last TC message Tuple: (src_addr, MPR_selector_set, MPR_set, T_time)*, where *src_addr* is the main address of the originator of the TC message, and *T_time* specifies the time when the tuple expires and must be removed from the set.

TC message reception: Upon receiving a TC message, a node will at first check the timestamp and the signature to see the freshness, the authentication and the integrity of the message. If any of the verifications fails, the message is discarded. If all the verifications succeed, the message is accepted and stored in the *Last Received TC Message Set* and waits to be processed. All TC messages stored longer than one TC_INTERVAL will be deleted from the set.

TC message processing

We now illustrate and detail the algorithm of TC message processing in TCSec. We suppose that a TC message TC^B generated by node B is received by node A:

1. For each node n_i in the *MPR Selector Set* of TC^B, A finds the last TC message(s) TC^{n_i} sent by n_i in its *Last Received TC Message Set*. If B is found in the MPR set of TC^{n_i}, A updates its *Topology Set* with the *Topology Tuple* (n_i, B, T_seq, T_time), where T_seq is set to the *Advertised Neighbor Sequence Number* (ANSN) of TC^B, and T_time is the expiration time of the tuple.

2. For each node n_j in the *MPR Set* of TC^B, A finds the last TC message(s) TC^{n_j} sent by n_j in its *Last received TC messages Set*. If B is in the *MPR Selector Set* of TC^{n_j}, A updates its *Topology Set* with the *Topology Tuple* (B, n_j, T_seq, T_time), where T_seq is set to the ANSN of TC^B, and T_time is the expiration time of the tuple.

3. For each *Topology Tuple* (T_dest_addr, T_last_addr, T_seq, T_time) in the *Topology Set* of A,

(a) If T_dest_addr is B, then
   i. If T_last_addr is not in the *MPR Selector Set* in TC^B, A removes the tuple.
   ii. Otherwise, A updates the *Topology Tuple* by resetting the validity time.

(b) If T_last_addr is B, then
   i. If T_dest_addr is not in the *MPR Set* in TC^B, A removes the *Topology Tuple*.
   ii. Otherwise, A updates the *Topology Tuple* by resetting its validity time.

6.4.3.4 Security analysis

Attack	TCSec	ADVSIG
Forged routing control message	OK	Partial
Control message replay	OK	OK
Colluding attacks	No	No
Selfish behavior	Better	No
Message relay	No	No

Table 6.1: Security analysis of TCSec

In this section we perform a security analysis for TCSec (for this we suppose the utilization of HPLS for HELLO messages). Table 6.1 illustrates the security features of TCSec compared to ADVSIG. "OK" in the table means that attack can be countered, and "No" indicates the contrary.
In the following, we discuss each of the attacks in details.

Incorrect control message generation

In TCSec, TC messages are protected by a global signature and a coherence check. An attacker cannot forge coherent declarations of MPR relationships except if it colludes with another compromised node. When there are two colluding compromised nodes, they can only establish forged MPR relationships between themselves. Therefore, under our assumptions, incorrect MPR information cannot be injected into the network.
However, an attacker can refuse to declare certain MPR or MPR selector information in its TC messages. This attack can invalidate some MPR relationships connected to the attacker, but it can also isolate the attacker.

Control message replay

Since each message is signed with a timestamp, the replay of an out-of-date OLSR routing message will be detected by the freshness check.

Colluding attacks

Colluding attacks such as wormhole attacks are always possible with both ADVSIG and TCSec. The attacks can make forged link/neighbor information between attackers be accepted by the other nodes.

Selfish behaviors

Since the sending of TC messages is necessary for the declaration of MPR nodes, selfish nodes cannot refuse to send TC. However, with TCSec we are still not able to fight against selfish nodes which declare no willingness to be MPR.
Moreover, to be selfish nodes have another possibility, which is to reduce (even to 0) the number of MPR Selector nodes in their TC messages. Note that in order not to be isolated, selfish nodes will not reduce the number of MPR nodes in their TC messages.

Message relay

In the following, we show an example of the relay attacks that cannot be countered by the TCSec scheme. We suppose that there are symmetric links $A \leftrightarrow X$ and $X \leftrightarrow B$, and A and B cannot hear each other:

$$A \leftrightarrow X \leftrightarrow B$$

If X is an attacker, it can misbehave by relaying all control messages and data traffic between A and B. This attack will make A and B believe that there is a symmetric link between them.
The relay attacks are difficult to prevent or detect. As in the mechanisms introduced in section 3.4.4.1, we may need strict time information or geographical information (for example, with the GPS module) to detect them.

6.4.3.5 Summary

We believe that TCSec has the following advantages compared to HPLS:

- It requires less cryptographic operations.
- It reinforces the cooperation from selfish nodes.

However, it also has the following disadvantages compared to HPLS:

- It may introduce a delay in verifying MPR relationships, since to verify a MPR relationship between nodes A and B, two TC messages TC^A and TC^B should both be received.
- There will be more TC messages sent in the network.

6.4.4 Simulation

The simulations are carried out under NS-2.28 [pro98]. TCSec is implemented by modifying the OLSR implementation named UM-OLSR that is provided by the University of Murcia [Ros]. We do not use multiple OLSR interface nodes in our simulations, since only single-interface nodes are configured in UM-OLSR.

6.4.4.1 Simulation setup

Parameter	Value
Simulation time	250s, 20s of initialization period
Field range	300m × 1500m
Number of nodes	30
Propagation model	Two-way ground
Power range	250m
Mobility model	Random way point
Mobility	Low - 2m/s as maximum speed Medium - 5m/s as maximum speed High - 20m/s as maximum speed
Pause time	5s
MAC protocol	IEEE802.11
MAC queue size	50
Traffic type	CBR 10 pkt/s
Number of flow	20
Packet size	64 bytes

Table 6.2: Simulation model and parameters

Parameter	Value
HELLO interval	2s
TC interval	5s
Holding time of neighbor information	6s
Holding time of topology information	15s

Table 6.3: OLSR setting

Table 6.2 shows our simulation model and parameters. Table 6.3 and 6.4 show respectively the OLSR setting and the cryptographic operation parameters in our simulations.
As also mentioned in chapter 5, we found that even though we can call cryptographic functions (thanks to external cryptographic libraries) in our simulations, the time of cryptographic computations that will definitely influence the real network performance is not counted by the simulator.

Parameter	Value
Signing delay	Randomly chosen from [0.2ms, 150ms], but stable for each node
Verification delay	Randomly chosen from [0.1ms, 100ms], but stable for each node
Signature length	320 bits
Hash operation delay	0.002 × Signing delay
Hash element length	128 bits

Table 6.4: Cryptographic operation parameters

Therefore in this chapter, in order to simulate the impact of the cryptographic operations on the performance of OLSR, we add some delay before processing or sending each HELLO or TC message. This delay is set according to [Raf05] which provides a suite of benchmarks for different cryptographic operations.
In addition, we also take into consideration the heterogeneity of nodes in ad hoc networks. We simulate nodes having different processing capacity: each node will have a random but fixed processing time for each cryptographic operation. For example, a signature computational delay ranges in (0.2ms, 150ms) and a signature verification delay ranges in (0.1ms, 100ms)[11].
In order to reduce the delay, in ADVSIG the ADVSIG messages are sent immediately after the sending of their corresponding routing messages. In our simulations we assume that there is no delay between the arrival of an original OLSR message and an ADVSIG message, thus all signatures can be immediately verified.

6.4.4.2 Performance simulation

In this section, we compare the performance of the scheme - HPLS for HELLO messages and TCSec for TC messages - to ADVSIG, We illustrate the performance improvements with relation to the replacement of digital signatures by Hash values and the TC coherence check. We simulate the following three metrics:

Control traffic overhead: The total overhead of TC messages and HELLO messages generated and relayed in the network, including the *proof*s and *certificate*s.

Data packet delivery ratio: The ratio of data packets generated by the CBR sources that are delivered to the destination.

Average end-to-end delay of data packets: The average delay between the emission of data packets by CBR sources and their arrival at destination.

[11] To add a delay before sending a message, it is sufficient to add it into the NS scheduler. As for the delay before processing a message, we can add a new timer to each routing information tuple which specifies the start time of the validity of the tuple.

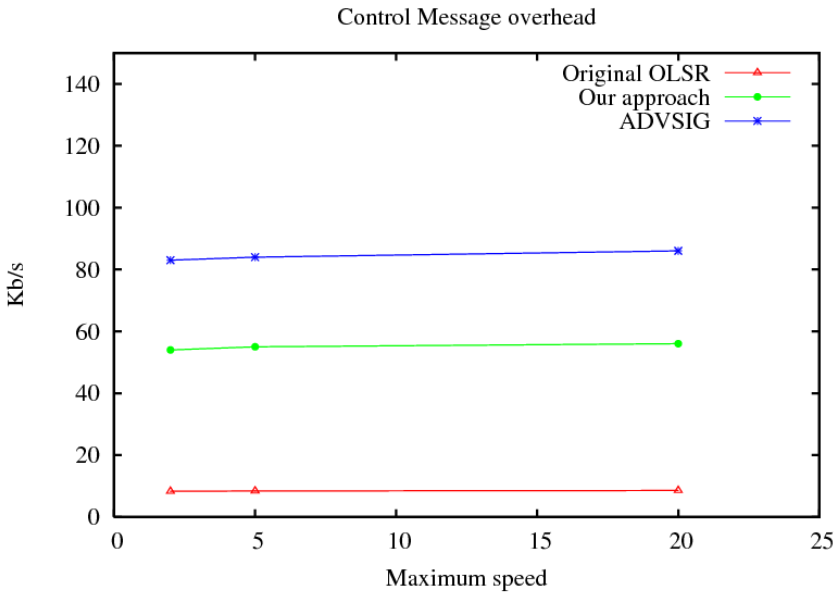


Figure 6.9: Control message overhead

Control message overhead

In ADVSIG, the additional control message overhead is mainly due to *proofs*, *certificates* and timestamps in TC and HELLO messages. In our scheme, the additional overhead is mainly due to the Hash values in HELLO messages, the *MPR Set* in TC messages and the additional TC messages.
Figure 6.9 compares the control message overhead of the original OLSR, ADVSIG and our scheme. We can see that, compared to the standard OLSR, the overhead in ADVSIG or in our scheme is significantly higher. However, our scheme generates much less control message overhead than ADVSIG.

Packet delivery ratio

We also compare in figure 6.11 the delivery ratios of the three approaches. Compared to the standard OLSR, our approach only degrades 5%-8% the delivery ratio, but ADVSIG degrades significantly the performance of OLSR. The result can be explained as follows:

For HELLO message reception: Suppose that q is the number of *proofs* to be verified in one HELLO message, then the typical value of q will range between 4 and 8. In ADVSIG, $q + 1$ verifications of *proof*s and one verification of *Global Signature* are to be performed when processing a HELLO message. Suppose that k HELLO messages should be processed by one node in one HELLO interval, then a node should in total perform $k \times (q + 1)$ signature verifications.

For a node having the verification processing time of 50ms, it needs on aver-

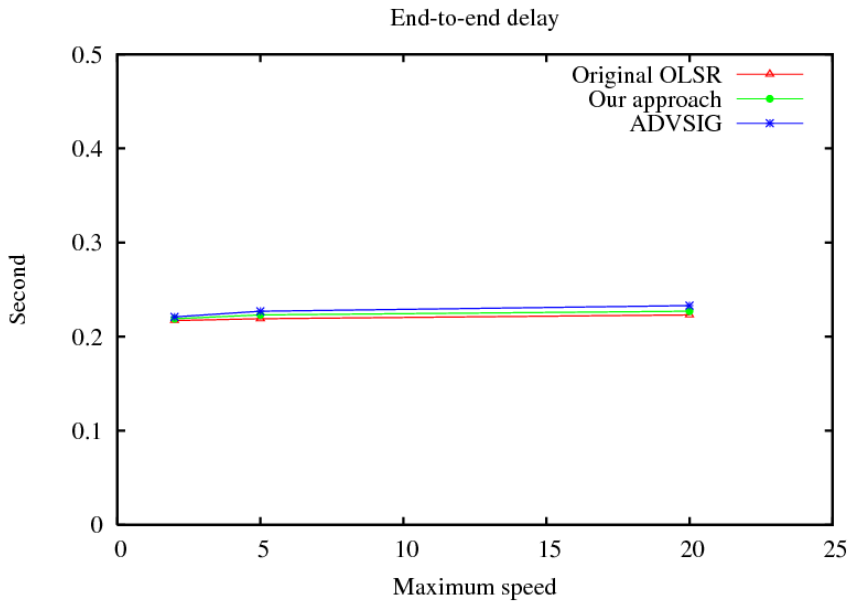


Figure 6.10: Packet delivery delay

age 3.5 seconds to perform all the signature verifications caused by HELLO messages (let alone the processing time of TC messages). As a result, it is not surprising that some nodes have no time to update their routing tables, and the log of NS always shows "no route available" when a data packet is lost.

In our approach, only some Hash operations and one signature verification (for the field *Global Signature*) are to be performed when processing a HELLO message. Performing substantially less verifications of signature, our approach shows better performance in the network where nodes have limited processing capacity.

For TC message reception: Suppose that p is the number of nodes in *MPR Selector Set* of TC messages, then the typical value of p ranges between 5 and 7. In ADVSIG, one verification operation is needed to check the validity of each *proof*, thus $p + 1$ signature verifications are to be performed when receiving a TC message. In a network of N nodes, in one TC interval, a node should perform $(p + 1) \times N/2$ signature verification operations under the condition that on average half of the nodes generate TC messages. As a result, when the verification delay or N increases, the performance of ADVSIG drops rapidly.

In our approach, in a N-node network only N signature verifications are to be performed by each node in each TC interval.

All two above reasons decide that weak capacity nodes in ADVSIG have not enough time to finish their cryptographic operations in each time interval, thus their routing tables cannot be updated, and the routes passing through them cannot be established.

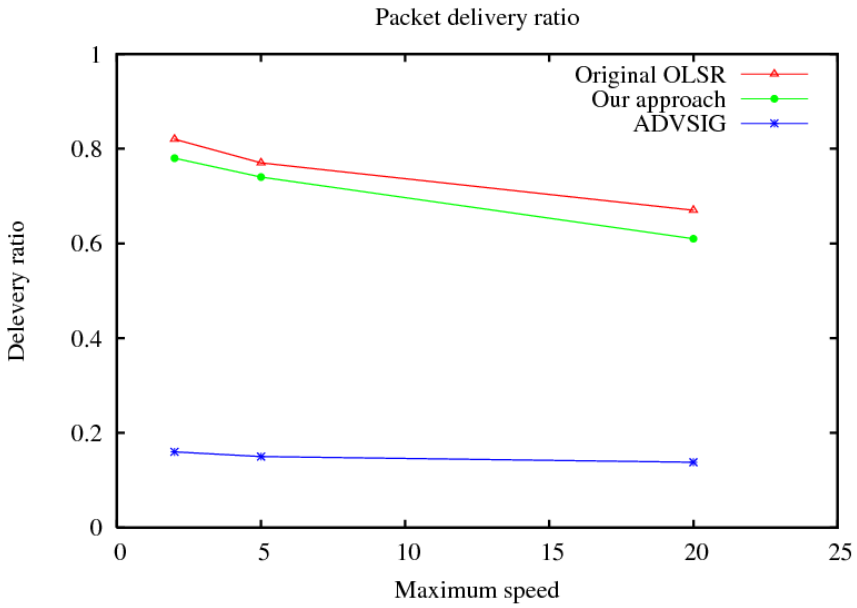


Figure 6.11: Packet delivery ratio

Average end-to-end delay of data packets

Figure 6.10 compares the average end-to-end delay of data packets. Our approach slightly outperforms ADVSIG without interfering the performance of OLSR.

6.5 Further discussion

In this section, we discuss the possibility of adding some supplementary but independent mechanisms to complete the security solutions proposed in this chapter. We also discuss the collision probability of Hash chain elements in HPLS. For a discussion regarding the synchronization, readers can refer to section 4.5.

6.5.1 MPR selection

In our mechanisms, MPR nodes are selected among neighbors with regard to their willingness to be MPR and their 2-hop neighbors. From the point of view of security, this may generate some vulnerabilities. A malicious node can show a high willingness to be MPR, in order to be selected as MPR node and then misbehave. Therefore, it is necessary to take the security into account in the MPR selection.
One possible solution consists of implementing a watchdog (c.f. chapters 4 and 5) on each node, in order to observe the behaviors of the neighbor nodes. Many attacks can thus be detected, such as greyhole, blackhole, modification of data packet header, etc. Then the results of the observations can be used as a criterion in the MPR selection: only benign nodes can be selected as MPR.
Different to TRP (c.f. chapter 5), here all the observations and decisions can be

made locally, and there is no need to exchange reputation values between nodes. And since there is no exchange of reputation, the blackmail attacks cannot exist neither.
Moreover, SWAN (c.f. chapter 4), if used, is only needed in the data forwarding phase, since the authentication of routing control messages can be guaranteed by the security mechanisms discussed in this chapter.

6.5.2 Redistribution of Hash chains

In HPLS, the server may need to redistribute the Hash chains when they are used up or when the number of nodes in the network exceeds the estimated upper bound. In the formal case, all Hash chains should be redistributed, while in the latter case, only the information about Hash chains related to the new nodes needs to be redistributed. We suggest overestimating the upper bound of the number of nodes and the lifetime of the ad hoc network to avoid frequently redistributing the Hash chains.

However, to resolve the problem of exhausted Hash values, we have an alternative choice which is much easier to be realized. Nodes can generate themselves enough Hash chains and use TC messages to broadcast the last values of the chains to the network. Since TC messages are signed, the last values are authenticated.

6.5.3 Collision probability of Hash chain elements

In HPLS, collisions of Hash chain elements may cause a security flaw. That is to say that different states of different links at different time intervals can correspond to a same *proof*. Hereby we perform an analysis on the Hash value collision probability $Prob(collision)$.
Let W be the number of Hash chain elements in the element space, then for l-bit Hash values, $W = 2^l$. Let U be the upper bound of the number of nodes in the network, $\mathfrak{L}$ be the length of Hash chains, then we note $Q = \mathfrak{L} \times (3U^2 - 3U)$ as the number of Hash chain elements generated by the server. We then have:

$$\begin{aligned}
\text{Prob(collision)} &= 1 - \text{Prob(no collision)} = 1 - \frac{{}^{Q}P_Q \times^{W} C_Q}{W^Q} \\
&= 1 - \frac{W!}{(W-Q)!W^Q} = 1 - \frac{W-1}{W}\frac{W-2}{W}\cdots\frac{W-(Q-1)}{W} \\
&= 1 - (1 - \frac{1}{W})(1 - \frac{2}{W})...(1 - \frac{Q-1}{W}) \\
&< 1 - (1 - \frac{Q-1}{W})^{Q-1} \sim 1 - (1 - \frac{(Q-1)^2}{W}) \sim \frac{Q^2}{W}
\end{aligned}$$

For example, in a 1000-node ad hoc network where each Hash chain contains 10^6 128-bit hash elements, we have $Prob(collision) < 10^{-28}$, which can be regarded as negligible.

6.5.4 Threat tree of HPLS and TCSec

The threat tree (c.f. chapter 2) of the approaches is shown in figure 6.12. We see from this figure that our approaches together with the MPR selection mechanism can protect the OLSR protocol from a large number of attacks.

6.6 Conclusion

In this chapter, we showed that both security mechanisms and their impacts on the network (routing) performance should be carefully taken into consideration when designing secure MANET routing protocols. We also proposed two lightweight and robust schemes, respectively Hash Proved Link State (HPLS) and Securing TC (TCSec), to secure the OLSR protocol.
As a first step, we analyzed a security flaw that is found in an existing secure OLSR mechanism named ADVSIG. To avoid the flaw, we then introduced an improved version of ADVSIG called ADVSIG$^+$, wherein we added an additional "*proof*" to asymmetric links. We also analyzed the overhead of ADVSIG to show that it may be too heavy to meet the performance requirements of the resource-restrained nodes in MANETs.
Then we developed a first approach named HPLS which secures HELLO and TC messages with Hash values. Both the computational overhead and the routing overhead of ADVSIG are then improved by HPLS, since a Hash value length is usually less important than the length of a digital signature, and Hash operations are certainly much more efficient than asymmetric cryptographic computations. However, these improvements are at the price of an offline server and a number of Hash chains to be stored by each node. We argued later with our simulations that this price is reasonable and overpaid by the improvements in performance.
We also proposed a mechanism named TCSec to secure TC messages in OLSR, which checks the coherence of MPR relationships from their both ends. Compared to ADVSIG, TCSec also substantially reduces the number of digital signatures to be computed and verified, and it also has the advantage of having less routing overhead. Both solutions are able to achieve the same security level as ADVSIG$^+$ and show better performance especially in the networks with resource-constrained nodes. The simulations done in the context of this research confirmed this fact.
In our simulations, we also found that ADVSIG can hardly achieve a good routing performance due to its large number of asymmetric cryptographic operations. Thus, we draw the conclusion that it is important that security mechanisms for MANETs are both robust and lightweight.

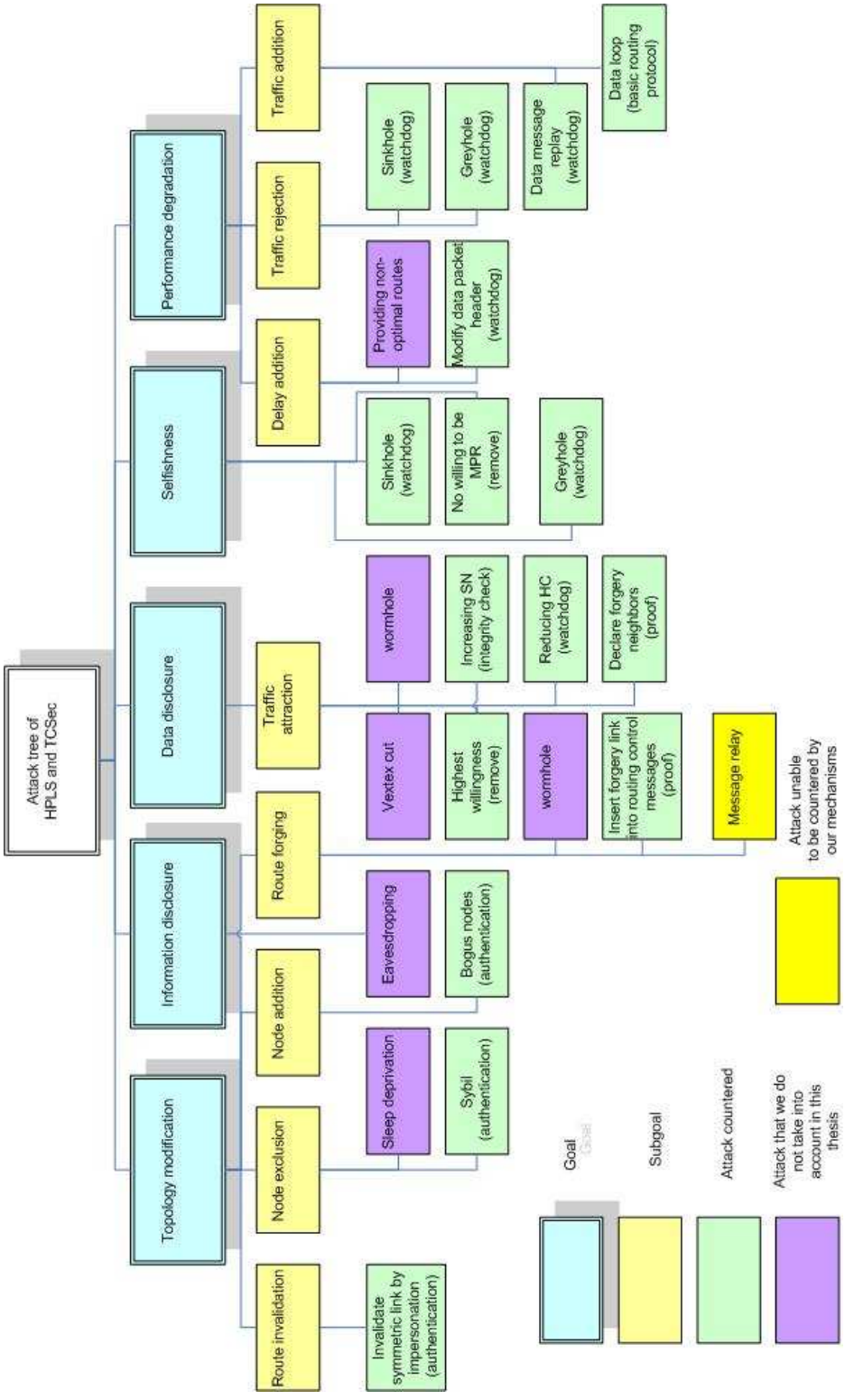


Figure 6.12: Threat tree of HPLS and TCSec

Chapter 7

Conclusion and future work

"An army burning with righteous indignation is bound to win."

– Lao zi (about 500 B.C.)

7.1 Conclusion

Mobile ad hoc networks present some particularities such as the mobility, the wireless interface, the independence to any infrastructure, etc.. More especially, it has an uncommon routing layer, thus new routing protocols should be designed. However, difficult challenges exist in the design of MANET secure routing protocols, since they should not only be robust against various attacks but also be efficient in terms of routing performance.

In this thesis, we have at first given an overview of ad hoc networks. We then classified all the known threats against the routing layer of MANETs into an attack tree. In this classification, we distinguished the attack objectives and the attack mechanisms, which will permit us to determine the attacks to counter under each security objective. Moreover, we also identified the new vulnerabilities raised due to the introduction of security mechanisms. Finally, we instantiated the attack tree for two existing routing protocols, namely DSR and OLSR, to show their vulnerabilities. In view of the tree, we also identified the attacks and misbehaviors that we will take into account in the design of our own secure routing mechanisms.

We have presented a state-of-art of security mechanisms designed to secure the ad hoc routing. The different mechanisms range from key establishment to selfish behavior prevention. We noted that, first, there are some mechanisms frequently used in a lot of security solutions, such as hash chains, watchdog, reputation system, etc., for the purpose of fighting against external attackers as well as compromised nodes. Secondly, some of the mechanisms are at a price of significant traffic and computational overhead, which is undesirable for the ad hoc networks having limited bandwidth and processing power. Thirdly, to satisfy the different requirements of different ad hoc scenarios and applications, MANETs need both reactive and proactive routing protocols, which led us to study the security for both the reactive and the proactive routing.

We then suggested a secure watchdog mechanism for ad hoc networks named SWAN, in which we blended SUCV and TESLA to develop a lightweight broadcast message authentication scheme. SWAN can be used to reduce the storage requirements of watchdog, and to prevent spoofing attacks that may badly affect the reputation systems. Our analysis showed that SWAN is both lightweight and robust, thus fulfilling the objectives of this research. Moreover, SWAN suits well the routing protocols based on the source routing algorithm, where packets are perfectly predictable.
We also proposed the Trust-based Routing Protocol (TRP). TRP is a reactive secure routing protocol based on the source routing algorithm and a reputation system. We secured the two routing phases in TRP: the routing discovery phase and the data delivery phase. In the first phase, we used an HMAC to protect routing control messages from end to end, and a watchdog to supervise the attacks that may take place in the middle of the routes. In the second phase, the watchdog is also used to supervise the attacks and misbehaviors committed on data packets. The reputation system based on the watchdog will give a rating to each route, and then source nodes will be able to choose the most reliable routes to send their data packets. By integrating reputation exchanges into routing control messages, we proposed the reactive reputation exchange, which permits to reduce the reputation exchange overhead and to protect the reputation exchanges against attacks while still take advantage of second-hand reputations. However, TRP is more suitable to ad hoc networks with a long lifetime and a frequent changing topology due to the reputation training phase.
At last, we studied the security issues in OLSR. We first analyzed a security flaw that we have found in ADVSIG and that allows an attacker to declare an asymmetric link as a symmetric link, and then we proposed an improved ADVSIG named ADVSIG$^+$ to counter this security flaw. In addition, we found that ADVSIG generates a high computational overhead and routing overhead, which will cause serious performance degradation in the ad hoc routing. Therefore we proposed two lightweight solutions for the security of OLSR, which will permit to reduce the lengths of message headers and the delay that is required to perform the cryptographic calculations. The first approach named HPLS uses Hash values to prove the link state of each link information in HELLO and TC messages, and the second solution named TCSec checks the coherence of MPR relationships by appending an additional header to each TC message. Simulations showed that our approaches outperform ADVSIG especially in the networks with limited bandwidth and processing power.

7.1.1 Guidelines on the design of a new ad hoc routing protocol secured from scratch

In this thesis, we have studied existing security mechanisms for ad hoc routing protocols, and we have also proposed several new mechanisms to secure the watchdog mechanism, the DSR reactive routing protocol and the OLSR proactive routing protocol. Moreover, in all our propositions, we treated performance issues as well as security issues.
However, securing an existing routing protocol might not be the best way to secure

MANETs. It may be better to design a new ad hoc routing protocol secured from scratch.
With the lessons obtained in this thesis, we believe that such a protocol should have the three basic elements: secure neighbor detection, authenticity of routing information, and the countermeasures against compromised nodes.
For the three points, we have considerations and further propositions as follows:

- Due to mobility, the *secure neighbor discovery* is very useful for ad hoc networks. A reliable neighbor detection mechanism can be the base of the MANET secure routing protocols since it can be useful for many other mechanisms such as supervision, topology discovery, route maintenance, wormhole prevention, etc... However, due to the relay attacks and the selfish behaviors, it is generally difficult to be sure of the neighborhoods without sophistic mechanism.

  To secure the neighbor detection, we have some possibilities such as GPS, OLSR secure neighbor discovery, SECTOR (c.f. section 3.4.4.1), etc.. Nevertheless, to be independent of any additional module or central server and to be more efficient, we can use the OLSR secure neighbor discovery process modified to remove the use of the MPR technique. Here is an example of the process which establishes a symmetric neighbor relationship between nodes A and B (all the notations can be found in section 6.2):

  1. $A \rightarrow B : \{\phi,\ \phi,\ T_A(t_0)\}_A$
  2. $B \rightarrow A : \{\text{“}A : ASYM\_LINK\text{”}, h(h(HELLO^A), B),\ T_B(t_1)\}_B$
  3. $A \rightarrow B : \{\text{“}B : SYM\_LINK\text{”}, h(h(HELLO^B), A),\ T_A(t_2)\}_A$

  In addition, we could use the promiscuous mode to check if a neighbor discovery message is relayed.

  In this scheme, since we no more need to use MPR, we can use hash values (but not hash chains) to completely replace the *proofs* in ADVSIG. The new scheme does not depend on any offline or online server, and is able to avoid the hash chain storage required by HPLS. The security level of this scheme is the same as ADVSIG$^+$. That is to say that it can prevent the link spoofing attacks committed by compromised nodes.

- Once we can be sure of neighborhood, the endairA protocol (c.f. section 3.4.2.8) may be used to ensure the *routing authenticity* while still keeping a good routing performance.

  Otherwise, if we want a proactive solution, we may use a protocol based on the distance-vector algorithm such as DSDV, and its security mechanism could be similar to the *coherence check* that is employed in TCSec. For example, if a node A declares that another node B is at i-hop away, then to validate the link node B should also declare that it needs i hops to reach A. Note that since we supposed that neighbor relationships are already secured, we can require

that only the symmetric links can be used in the routing, thus no incoherence can be caused by asymmetric links.

To improve the efficiency in the reactive route hunting, we can think of another optimization with the help of a reputation system. If we can guarantee the security of the RREQ messages, a destination node can wait for a timeout before sending back multiple routes in one RREP message if it received multiple RREQs during this timeout. The RREP can itself use the most reliable route (according to the destination node) among all collected routes (to be safer, the RREP may also use a limited number of routes). This mechanism has two advantages. First, a single RREP will permit the source node to register many routes and to choose one of the best routes for its data delivering (as in TRP, the reputation system can help the source node to make choice between a set of routes). Thus the routing overhead is reduced compared with sending multiple RREPs. Second, the rushing attacks (c.f. section 3.4.4.3) can also be naturally prevented by this mechanism. However, this mechanism cannot be directly applied to TRP, since TRP does not guarantee the security of RREQs. Only if we can correctly authenticate the intermediate nodes during the propagation of RREQs, the application of this mechanism becomes possible.

- If the integrity of routing information is ensured, we can focus on the security issues caused by *compromised nodes* especially in the data delivery phase. For this we may use watchdog or design a cross-layer solution.

  Until now, all the security solutions that we have discussed in this thesis are only at the routing layer. However, cross-layer security mechanisms may also worth be considered. For example, to secure the data delivery, we can design a secure transport layer protocol similar to TLS which takes into consideration the characteristics of MANETs to guarantee the end-to-end security proprieties for data flux. The main advantage would be that it is independent of the routing protocol in use.

In addition, we also present the following considerations that may be useful for the further improvements of secure routing in MANETs:

- Trust can play a more important role in the secure routing. Take the RAP protocol as example, instead of randomly choose a RREQ to rebroadcast, a node can choose a RREQ from a neighbor that it trusts.

- If trust can be transitive, in a proactive routing protocol we can let each intermediate node choose its next hop according to trust and route length.

After all, we believe that it is necessary to study case by case the scenarios of MANETs, in order to find out the best security solutions for each typical ad hoc scenario/application.

7.2 Future research directions

The work in this dissertation make us elaborate the following short-term research directions:

- The reputation system presented in chapter 5 deserves further studies. A formal analysis is necessary to theoretically demonstrate its correctness and efficiency. Further simulations can also be carried out to adjust the security parameters in TRP under each typical network scenario.

- For all the secure routing protocols proposed in this thesis, we need to formally validate their security proprieties. For example, a formal analysis will be appreciated if it can prove that no link spoofing attack can occur in HPLS and TCSec.

- In addition to SWAN, we can further look for the solutions for the hidden node problems that can negatively influence the supervision results.

We also underline some long-term research directions that require more investigations in the future:

- The attack tree presented in chapter 2 should be further enriched every time there is a newly found vulnerability against the MANET routing layer. It should also be completed and developed with attacks present in diverse routing protocols and security mechanisms. As a consequence, the trees presented later in sections 2.7, 5.6.4, 5.7.2 and 6.5.4 also need to be progressively updated.

- For the design of future secure routing protocols, we can provide to simulators, such as NS-2, GlomoSim [ZBG98], etc..., the capacity of taking the computational delays generated by cryptographic operations into account with simple configurations. We can also supply simulators the possibility to easily carry out simulations under hostile environment.

- Wherever there is control or communication exchange, there could be threats and attacks (not limited to the routing layer). The synchronization mechanisms, transport layer protocols, MAC layer protocols, etc., can also be targets of attacks. Therefore we can also consider the security issues on the other network layers of MANETs.

  For example, the MAC layer security could be an interesting room for creativity. The main weakness of the MANET MAC layer seen from the point of view of security is that the activities and the visions of nodes at the MAC layer are limited to their neighborhood due to the radio medium nature. Thus, the prevention of greedy behaviors is an important issue for the MAC layer security. Currently there are already a lot of propositions [KV05, RHA, CGAH04] aiming to reduce the greedy behaviors on the MAC layer of MANETs.

- Finally, we need to take into consideration the attacks that we mentioned as untreated in this thesis. Even though some attacks are difficult to realize from today's point of view, more investigations should be provided to them to face the most critical scenarios that could take place in the future, since all theoretical attack scenarios tend to become practical attacks sooner or later. Therefore, preparing for the worst will always give us an additional safety margin.

Bibliography

[ACL+05] Cédric Adjih, Thomas Clausen, Anis Laouiti, Paul Mühlethaler, and Daniele Raffo. Securing the OLSR routing protocol with or without compromised nodes in the network. Technical report, INRIA RR-5494. HIPERCOM project, INRIA Rocquencourt, France, February 2005.

[AG00] N. Asokan and Philip Ginzboorg. Key-agreement in ad-hoc networks. *Computer Communications*, 23(17):1627–1637, 2000.

[AHNRR02] Baruch Awerbuch, David Holmer, Cristina Nita-Rotaru, and Herbert Rubens. An on-demand secure routing protocol resilient to byzantine failures. In *ACM Workshop on Wireless Security (WiSe)*, Atlanta, Georgia, September 2002.

[AK96] R. Anderson and M. Kuhn. Tamper Resistance - a Cautionary Note. In *Proceedings of the Second Usenix Workshop on Electronic Commerce*, pages 1–11, November 1996.

[BB02a] Sonja Buchegger and Jean-Yves Le Boudec. Cooperative Routing in Mobile Ad-hoc Networks: Current Efforts Against Malice and Selfishness. In *Lecture Notes on Informatics, Mobile Internet Workshop, Informatik 2002*, Dortmund, Germany, October 2002. Springer.

[BB02b] Sonja Buchegger and Jean-Yves Le Boudec. Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks. In *10th Euromicro Workshop on Parallel, Distributed and Network-based Processing (Euromicro-PDP 2002)*, pages 403–410, Canary Islands, Spain, January 2002.

[BB03] Sonja Buchegger and Jean-Yves Le Boudec. The Effect of Rumor-Spreading in Reputation Systems for Mobile Ad-hoc Networks. In *WiOpt'03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks*, Nice, France, March 2003. INRIA.

[bbb05] Ieee standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks– specific requirements part 15.1: Wireless medium access control (mac) and physical layer (phy) specifications for wireless personal area networks (wpans), 2005.

[BBK94] Thomas Beth, Malte Borcherding, and Birgit Klein. Valuation of trust in open networks. In *Proceedings of the 3rd European Symposium on Research in Computer Security – ESORICS '94*, pages 3–18, 1994.

[BEGA02] Rakesh B. Bobba, Laurent Eschenauer, Virgil D. Gligor, and William Arbaugh. Bootstrapping security associations for routing in mobile ad-hoc networks. Technical report, Technical Report 2002-44, The Institute for Systems Research, 2002.

[BF01] Dan Boneh and Matthew K. Franklin. Identity based encryption from the weil pairing. *Advances in Cryptology - Crypto 2001, Lecture Notes in Computer Science 2139*, pages 213–229, 2001.

[BGLS03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. *Proceedings of EUROCRYPT*, 2003.

[BH01] Levente Buttyán and Jean-Pireer Hubaux. Nuglets: a virtual currency to stimulate cooperation in self-organized mobile ad hoc networks. Technical report, Technical Report DSC/2001/001, Swiss Federal Institute of Technology - Lausanne, 2001.

[BHBR01] Stefano Basagni, Kris Herrin, Danilo Bruschi, and Emilia Rosti. Secure pebblenet. In *Proceedings of the 2001 ACM Iternational Symposium on Mobile Ad Hoc Networking & Computing, MobiHoc 2001*, pages 156–163, Long Beach, CA, USA, October 4–5 2001.

[BSSW02] Dirk Balfanz, Diana Smetters, Paul Stewart, and H. Chi Wong. Talking to strangers: Authentication in adhoc wireless networks, 2002.

[BV04] Levente Buttyán and István Vajda. Towards provable security for ad hoc routing protocols. Technical report, Technical Report, Budapest University of Technology and Economics, July 2004.

[CB05] Thomas Clausen and Emmanuel Baccelli. Securing olsr problem statement, draft-clausen-manet-solsr-ps-00.txt, February 2005.

[CBH03] Srdjan Capkun, Levente Buttyan, and Jean-Pierre Hubaux. Sector: Secure tracking of node encounters in multi-hop wireless netowrks. In *Proceedings of the 1-st ACM Workshop on Security of Ad Hoc and Sensor Networks (SANS 2003)*, pages 21–32, Fairfax, Virginia, USA, 2003.

[CEH+] Mark J. Cox, Ralf S. Engelschall, Stephen Henson, Ben Laurie, and al. OpenSSL project, http://www.openssl.org/.

[CGAH04] Mario Cagalj, Saurabh Ganeriwal, Imad Aad, and Jean-Pierre Hubaux. On cheating in csma/ca ad hoc networks. Technical report, Technical Report, École polytechnique Fédérale de Lausanne, March 2004.

[Cha96] D. Chaum. Private signature and proof systems. *United States Patent 5493614*, 1996.

[Cha04] Aldar C-F. Chan. Distributed symmetric key management for mobile ad hoc networks. In *IEEE INFOCOM'04 - The Conference on Computer Communications*, volume 23, pages 2414–2424, March 2004.

[CHB03] Srdjan Capkun, Jean-Pierre Hubaux, and Levente Buttyán. Mobility helps security in ad hoc networks. In *Proceedings of the ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2003)*, pages 46–56, June 2003.

[Che97] Steven Cheung. An efficient message authentication scheme for link state routing. In *Annual Computer Security Applications Conference (ACSAC 1997)*, pages 90–98, San Diego, CA, USA, December 1997.

[CJ03] T. Clausen and P. Jaquet. Rfc 3626 - optimized link state routing protocol, October 2003.

[CM99] S. Corson and J. Macker. Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations, January 1999. RFC 2501.

[CPS03] Haowen Chan, Adrian Perrig, and Dawn Song. Random key predistribution schemes for sensor networks. In *2003 IEEE Symposium on Research in Security and Privacy*, 2003.

[CXL06] Lin Chen, Xiaoyun Xue, and Jean Leneutre. A lightweight mechanism to secure olsr. In *Proceedings of the International MultiConference of Engineers and Computer Scientists 2006*, Hong Kong, June 2006.

[CY02] Stephen Carter and Alec Yasinsac. Secure position aided ad hoc routing protocol. In *proceedings of the IASTED International Conference on Communications and Computer Networks (CCN02)*, November 2002.

[DH98] S. Deering and R. Hinden. Internet protocol, version 6 (ipv6) specification, December 1998. RFC 2460.

[DMMJ98] D.Maughan, M.Schertler, M.Scheneider, and J.Turner. Internet security association and key management protocol (isakmp), November 1998. RFC 2408.

[Dou02] John R. Douceur. The sybil attack, March 2002.

[Dro97] R. Droms. Dynamic host configuration protocol (DHCP), March 1997. RFC 2131.

[EG02] Laurent Eschenauer and Virgil D. Gligor. A key management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communications security*, pages 41–47, November 2002.

[FKK96] Alan O. Freier, Philip Karlton, and Paul C. Kocher. The SSL protocol version 3.0, November 1996. INTERNET DRAFT, work in progress. draft-freier-ssl-version3-02.txt.

[FT91] D. Fudenberg and J. Tirole. *Game Theory*. The MIT Press, Cambridge, Massachusetts, 1991.

[GZ02] Manel Guerrero Zapata. Secure ad hoc on-demand distance vector (SAODV) routing. First published in the IETF MANET Mailing List (October 8th 2001), August 2002. INTERNET-DRAFT draft-guerrero-manet-saodv-00.txt, work in progress.

[HBC01] Jean-Pierre Hubaux, Levente Buttyán, and Srdjan Capkun. The quest for security in mobile ad hoc networks. In *Proceedings of ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC 2001)*, pages 255–265, Long Beach, CA, USA, October 4-5 2001.

[HC98] D. Harkins and D. Carrel. The internet key exchange (IKE), November 1998. RFC 2409.

[HHF05] Fan Hong, Liang Hong, and Cai Fu. Secure olsr. In *Advanced Information Networking and Applications (AINA 2005)*, volume 1, March 2005.

[Hie01] Maarit Hietalahti. Key establishment in ad-hoc networks, 2001.

[HJP02] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. Sead: Secure efficient distance vector routing in mobile wireless ad hoc networks. In *Fourth IEEE Workshop on Mobile Computing Systems and Applications* (WMCSA '02), pages 3–13, Callicoon, NY, USA, June 2002.

[hNC] IETF MANET (Mobile Ad hoc NETworks) Charter. http://www.ietf.org/html.charters/manet-charter.html.

[HPB03] Yih-Chun Hu, Adrian Perrig, and David B.Johnson. Rushing attacks and defense in wireless ad hoc network routing protocols. In *Proceedings of the 2003 ACM workshop on Wireless security*, pages 30–40, 2003.

[HPJ02] Y. Hu, A. Perrig, and D. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *The 8th ACM International Conference on Mobile Computing and Networking (ACM MobiCom 2002)*, pages 12–23, Atlanta, Georgia, USA, September 2002.

[HPJ03] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *INFOCOM*, April 2003.

[HPS02] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. The zone routing protocol (zrp) for ad hoc networks, July 2002. INTERNET-DRAFT draft-ietf-manet-zone-zrp-04.txt, work in progress.

[HPT99] Ralf Hauser, Tony Przygienda, and Gene Tsudik. Lowering security overhead in link state routing. *Computer Networks (Amsterdam, Netherlands)*, 31(8):885–894, 1999.

[HWD02] H.Deng, W.Li, and D.P.Agrawal. Routing security in wireless ad hoc networks. *IEEE Communications Magazine*, 40(10):70–75, October 2002.

[IN83] K. Itakura and K. Nakamura. A public key cryptosystem suitable for digital multisignatures. *NEC Research & Development*, 71:1–8, 1983.

[iso94] Information technology - open systems interconnection - basic reference model: The basic model, 1994. ISO/IEC 7498-1:1994(E).

[JAA04] Ning Jiang, Kien A.Hua, and Mounir A.Tantaoui. Collaboration enforcement and adaptive data redirection in mobile ad hoc networks using only first-hand experience. In *Mobile and Wireless Communications Networks*, pages 227–238, October 2004.

[Jak02] Markus Jakobsson. Fractal hash sequence representation and traversal. In *IEEE International Symposium on Information Theory (ISIT '02)*, pages 437–444, 2002.

[JMH04] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The dynamic source routing protocol for mobile ad hoc networks (DSR), July 2004. INTERNET-DRAFT draft-ietf-manet-dsr-10.txt, work in progress.

[JSI96] M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. *Proceedings of EUROCRYPT*, 1996.

[KBC97] H Krawczyk, M Bellare, and R Canetti. Hmac: Keyed-hashing for message authentication, Feb 1997. RFC 2104.

[KKA03] Aram Kahlili, Jonathan Katz, and William A. Arbaugh. Toward secure key distribution in truly ad-hoc networks. In *Proceedings of the IEEE Workshop on Security and Assurance in Ad-Hoc Networks*, pages 342–246, 2003.

[KV05] Pradee Kyasaur and Nitin H. Vaidya. Selfish mac layer misbehavior in wireless networks. *IEEE Transactions on mobile computing*, 4(5), September/October 2005.

[KZL+01] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *International Conference on Network Protocols (ICNP)*, pages 251–260, 2001.

[Lab02] RSA Laboratories. PKCS #1 v2.1: Rsa cyrptography standard, June 2002.

[Lam81] Leslie Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24, No. 11:770–772, November 1981.

[Len05] Arjen K. Lenstra. Further progress in hashing cryptanalysis, February 2005. Lucent Technologies, Bell Laboratories.

[LL00] Haiyun Luo and Songwu Lu. Ubiquitous and robust authentication services for ad hoc wireless networks. Technical report, TR-200030, Dept. of Computer Science, UCLA, 2000.

[LZ03] Ten-Hwang Lai and Dong Zhou. Efficient and scalable ieee 802.11 ad-hoc-mode timing synchronization function. In *Proceedings of the IEEE 17th International Conference on Advanced Information Networking and Applications (AINA'03)*, Xi'an, China, March 2003.

[Mar02] John Marshall. An analysis of SRP for mobile ad hoc networks. In *International Multi-Conference in Computer Science*, June 2002.

[MC02] Gabriel Montenegro and Claude Castelluccia. Statistically unique and cryptographically verifiable (sucv) identifiers and addresses. In *Proceedings of the 9th Annual Network and Distributed System Security Symposium (NDSS 2002)*, San Diego, California, USA, February 2002.

[Mer80] Ralph C. Merkle. Protocols for public key cryptosystems. In *SIMMONS: Secure Communications and Asymmetric CryptoSystems*, pages 122–134, 1980.

[MGLB00] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking (Mobicom 2000)*, pages 255–265, Boston, USA, 2000.

[Mic04] Pietro Michiardi. *Cooperation enforcement and network security mechanisms for mobile ad hoc networks.* PhD thesis, PhD Thesis of ENST Eurécom, December 2004.

[MM02] Pietro Michiardi and Refik Molva. Core: A COllaborative REputation mechanism to enforce node cooperation in mobile ad hoc networks. In *Communication and Multimedia Security 2002 Conference*, 2002.

[MM03] Pietro Michiardi and Refik Molva. Ad hoc network security. *BST Microelectronics Journal of System Research*, 2003.

[MM04] C. Siva Ram Murthy and B. S. Manoj. *Ad hoc Wireless Networks : Architectures and Protocols*. Prentice Hall Communications, 2004.

[MOR01] S. Micali, K. Ohta, and L. Reyzin. Accountable-subgroup multisignatures. *Proceedings of the 8th ACM Conference on Computer and Communications Security*, pages 245–254, 2001.

[Moy98] J. Moy. Rfc 2328 - ospf version 2, April 1998.

[NDBJ01] Andrew Nash, Bill Duane, Derek Brink, and Celia Joseph. *PKI: Implementing and Managing E-Security*. McGraw-Hill, 2001. ISBN 0072131233.

[Net97] Broadband Radio Access Networks. HIPERLAN type 2 system overview, 1997.

[Net98a] Broadband Radio Access Networks. HIgh PErformance radio local area network (HIPERLAN) type 1 functional specification, 1998.

[Net98b] Broadband Radio Access Networks. HIPERLAN type 2 requirements and architectures for wireless broadband access, 1998.

[NNS98] T. Narten, E. Nordmark, and W. Simpson. Neighbor discovery for ip version 6 (ipv6), December 1998. RFC 2461.

[NSSP04] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks: analysis & defenses. In *Proceedings of the third international symposium on Information processing in sensor networks*, Berkeley, California, USA, April 2004.

[oEE] Institute of Electrical and Electronics Engineers. Ieee. http://www.ieee.org/portal/site.

[Ohr05] Frank Ohrtman. *WiMAX Handbook*. McGraw-Hill Professional, 1st edition, May 2005. ISBN 0071454012.

[PB94] Charles Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[PBRD02] Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir R. Das. Ad hoc On-Demand Distance Vector (AODV) Routing, November 2002. INTERNET-DRAFT draft-ietf-manet-aodv-12.txt, work in progress.

[PCJS00] Adrian Perrig, Ran Canetti, J.D.Tygar, and Dawn Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, Oakland, CA, May 2000.

[PCSJ01] Adrian Perrig, Ran Canetti, Dawn Song, and J.D.Tygar. Efficient and secure source authentication for multicast. In *Network and Distributed System Security Symposium (NDSS'01)*, pages 35–46, February 2001.

[Per01] Charles Perkings, editor. *Ad Hoc Networking*. Addison-Wesley, 2001.

[PH02] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure routing for mobile ad hoc networks. In *Proceedings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, pages 27–31, San Antonio, TX, January 2002.

[PH03] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure link state routing for mobile ad hoc networks. In *Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, 2003.

[pro98] The VINT project. NS notes and documentation, May 1998. Work in progress.

[PSW+01] Adrian Perrig, Robert Szewczyk, Victor Wen, David E. Culler, and J. D. Tygar. SPINS: security protocols for sensor netowrks. In *Proceedings of the 7th annual international conference on Mobile computing and networking (ACM MOBICOM 2001)*, pages 189–199, Rome, Italy, 2001.

[PZ03] P.Papadimitratos and Z.J.Haas. Secure data transmission in mobile ad hoc networks. In *Proceedings of the 2003 ACM workshop on Wireless security table of contents*, pages 41–50, 2003.

[RACM04] Daniele Raffo, Cedric Adjih, Thomas Clausen, and Paul Muhlethaler. An advanced signature system for olsr. In *Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks (WiSe 2004)*, pages 10–16, Washington DC, USA, 2004.

[Raf05] Daniele Raffo. *Security Schemes for the OLSR Protocol for Ad hoc Networks*. PhD thesis, Thesis of University Paris 6 - Pierre et Marie Curie, September 2005.

[Res99] E. Rescorla. Diffie-hellman key agreement method, June 1999. RFC 2631.

[RHA] Maxim Raya, Jean-Pierre Hubaux, and Imad Aad. Domino: A system to detect greedy behavior in IEEE hotspots.

[RK04] Carlos H. Rentel and Thomas Kunz. Network synchronization in wireless ad hoc networks. Technical Report Technical Report SCE-04-08, Carleton University, Computer Engineering, July 2004.

[RLTN93] Version 1.4 RSA Laboratories Technical Note. PKCS #3 - diffie-hellman key agreement standard, November 1993.

[Ros] Francisco J. Ros. UM-OLSR: Main page, http://ants.dif.um.es/masimum/um-olsr/html/.

[RST01] R. Rivest, A. Shamir, and Y. Tauman. How to leak a secret. *Proceedings of Asiacrypt'01*, 2248:552–565, 2001.

[SA99] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols*, pages 172–194, April 1999.

[Sch00] Bruce Schneier. *Secrets and Lies. Digital Security in a Networked World.* John Wiley & Sons, Inc, 1 edition, 2000.

[SDL+02] Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of 2002 IEEE International Conference on Network Protocols (ICNP)*, Paris, France, November 2002. http://signl.cs.umass.edu/arand/.

[S.G95] S.Garfinkel. *PGP: Pretty Good Privacy.* O'Reilly & Associates, 1995. ISBN 1-56592-098-8.

[SH02] Christian Schwingenschlögl and Marc-Philipp Horn. Building blocks for secure communication in ad-hoc networks. In *Proceedings of the 4th European Wireless (EW'02)*, Florence, Italy, February 25-28 2002.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sho00] Victor Shoup. Practical threshold signatures. *Lecture Notes in Computer Science*, 1807:207–??, 2000.

[Soc05] IEEE Computer Society. Information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications, 2005.

[Sta01] Frank Stajano. The resurrecting duckling – what next? *Lecture Notes in Computer Science*, 2133:204–210, 2001.

[Ste06] Marc Stevens. Fast collision attack on md5, March 2006.

[SV04] Jungmin So and Nitin H. Vaidya. A distributed selfstabilizing time synchronization protocol for multi-hop wireless networks. Technical report, Technical Report, UIUC, January 2004.

[Uni06] Columbia University. Small world project, at http://smallworld.columbia.edu/, 2006.

[WHiKlS05] Eli Winjum, Anne Marie Hegland, Ø ivind Kure, and Pål Spilling. Replay attacks in mobile wireless ad hoc networks: Protecting the olsr protocol. In *International Conference on Networking (ICN 2005)*, Reunion Island, April 2005.

[WlSiK05] Eli Winjum, Pål Spilling, and Ø ivind Kure. Trust metric routing to regulate routing cooperation in mobile wireless ad hoc networks. In *the 11th European Wireless Conference*, Nicosia Cyprus, 10-13, April 2005.

[WY05] Xiaoyun Wang and Hongbo Yu. How to break md5 and other hash functions. *Eurocrypt*, pages 19–35, 2005.

[WZ02] K. Weniger and M. Zitterbart. Ipv6 autoconfiguration in large scale mobile ad-hoc networks. In *Proceedings of European Wireless 2002*, Florence, Italy, February 2002.

[XLB04] Xiaoyun Xue, Jean Leneutre, and Jalel BenOthman. A trust-based routing prtocol for ad hoc networks. In *Mobile and Wireless Communications Networks*, pages 251–262, October 2004.

[XLCBO06] Xiaoyun Xue, Jean Leneutre, Lin Chen, and Jalel Ben-Othman. Swan: A secured watchdog for ad hoc networks. *International Journal of Computer Science and Network Security*, 6(2), 2006.

[YB94] Raphael Yahalom and Thomas Beth. Trust relationships in secure systems - a distributed authentication perspective. *Computer Systems*, 7(1):45–73, 1994.

[YML02] Hao Yang, Xiaoqiao Meng, and Songwu Lu. Self-organized network layer security in mobile ad hoc networks. In *ACM MOBICOM Wireless Security Workshop (WiSe)*, Atlanta, September 2002.

[YNK02] Seung Yi, Prasad Naldurg, and Robin Kravets. A security-aware ad hoc routing protocol for wireless networks. In *The 6th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI 2002)*, 2002.

[ZA02] Manel Guerrero Zapata and N. Asokan. Securing ad-hoc routing protocols. In *Proceedings of the 2002 ACM Workshop on Wireless Security (WiSe)*, pages 1–10, September 2002.

[ZBG98] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. Glomosim: A library for parallel simulation of large-scale wireless networks. In *Workshop on Parallel and Distributed Simulation*, pages 154–161, 1998.

[ZCY03] Sheng Zhong, Jiang Chen, and Yang Richard Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2003.

[ZH99] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, November-December 1999.

[ZSvR02] Lidong Zhou, Fred B. Schneider, and Robbert van Renesse. COCA: A secure distributed on-line certification authority. *ACM Transactions on Computer Systems*, 20(4):329–368, November 2002.

[ZXSJ03] Sencun Zhu, Shouhuai Xu, Sanjeev Setia, and Sushil Jajodia. LHAP: A lightweight hop-by-hop authentication protocol for ad-hoc networks. In *ICDCS 2003 International Workshop on Mobile and Wireless Network (MWN 2003)*, May 2003.

Publications

International journal

[1] Xiaoyun Xue, Jean Leneutre, Lin Chen and Jalel Ben-Othman. SWAN: A Secured Watchdog for Ad hoc Networks, in *International Journal of Computer Science and Network Security*, Vol.6, Num.2, 2006.

International conferences

[2] Lin Chen, Xiaoyun Xue and Jean Leneutre. A Lightweight Mechanism to Secure OLSR. In *Proceedings of the International MultiConference of Engineers and Computer Scientists 2006*, Hong Kong, China. 20-22 June, 2006.

[3] Xiaoyun Xue, Jean Leneutre and Jalel Ben-Othman. A trust-based routing protocol for ad hoc networks. In *Proceedings of the 6th IFIP IEEE International Conference on Mobile and Wireless Communication Networks (MWCN'04)*, Paris, France. Octobre 2004.

[4] Jalel Ben-Othman and Xiaoyun Xue. SERAN : A new protocol to hide an equipment in ad hoc networks. In *Proceedings of the 8th IEEE symposium on computer and communications (ISCC'03)*, Kemer-Antalya, Turkey. Juillet 2003.

[5] Jalel Ben-Othman and Xiaoyun Xue. Security equipment in ad hoc networks. In *Proceedings of the IEEE Semi-annual Vehicular Technology Conference (VTC Spring 2002)*, Birmingham, Al, USA. Mai 2002.

National conferences

[6] Xiaoyun Xue, Jean Leneutre and Jalel Ben-Othman. Un protocole de routage ad-hoc basé sur un modèle de confiance distribué et valué. In *Proceedings of the 3ème conférence sur la sécurité et Architectures Réseaux (SAR'04)*, La Londe, Côte-d'Azur, France. Juin 2004.

[7] Jalel Ben-Othman and Xiaoyun Xue. Sécurisation d'un équipement dans un réseau ad hoc. In *Proceedings of Journées doctorales informatique et réseaux (JDIR 2002)*, Toulouse, France. Mars 2002.

Glossary

-A-	
ADVSIG	ADVanced SIGnature
AODV	Ad hoc On Demand Distance Vector
AMPS	Advanced Mobile Phone System
APS	Active Path Set
ARAN	Authenticated Routing for Ad hoc Networks
ARP	AutoRadioPuhelin, Car Radio Phone in English
-B-	
BT	Bluetooth network
BMA	Broadcast Message Authentication
BWA	Broad Band Wireless Access networks
-C-	
CA	Certificate Authority
CBR	Constant Bit Rate
CCS	Credit Clearance Service
CDMA	Code Division Multiple Access
CH	Cluster Header
COCA	Cornell On-line Certification Authority
CONFIDANT	Cooperation Of Nodes Fairness In Dynamic Ad-hoc NeTworks
CORE	COllaborative REputation
CPU	Central Processing Unit
CRL	Certificate Revocation List
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
-D-	
DCF	Distributed Coordination Function
DH	Diffie-Hellman
DHCP	Dynamic Host Configuration Protocol
DoS	Denial of Service
DN	Direct Neighbors
DSA	Digitial Signature Algorithm
DSDV	Highly Dynamic Destination-Sequenced Distance-Vector Routing
DSDV-SQ	DSDV for Sequence Numbers
DSR	Dynamic Source Routing protocol
DSSS	Direct Sequence Spread Spectrum

-E-

ECNR	Elliptic Curve Nyberg. Rueppel
ETSI	European Telecommunications Standards Institute

-F-

FHSS	Frequency Hopping Spread Spectrum

-G-

GPS	Global Position System
GSM	Global System for Mobile communications
GTE	Guaranteed Time of Encounter

-H-

HiperLAN	High Performance LAN
HMAC	keyed Message Authentication Code
HPLS	Hash Proved Link State

-I-

IARP	IntrAzone Routing Protocol
IBSS	Independent Basic Service Set
ID	IDentifier
IDHC	ID-based message authentication
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
INRT	Intermediate Node Reply Token
IP	Internet Protocol
IPv6	Internet Protocol version 6
ISO	International Standards Organization

-K-

KD	Key Disclosure message
KDC	Key Distribution Center
KREQ	Key REQuest
KREP	Key REPly

-L-

LHAP	Lightweight Hop-by-hop Authentication Protocol
LSU	Link State Update

-M-

MAC (1)	Media Access Control
MAC (2)	Message Authentication Code
MAD	Mutual Authentication with Distance-bounding
MANET	Mobile Ad hoc Network
MHT	Merkle Hash Tree
MPR	MultiPoint Relaying
MSN	Message Sequence Number

-N-

NDP	Neighbor Discovery Protocol
NLP	Neighbor Lookup Protocol
NS-2	Network Simulator 2

-O-

OFDM	Orthogonal Frequency Division Multiplexing
OLSR	Optimized Link State Routing protocol
OSI	Open Systems Interconnection Reference Model
OSPF	Open Shortest Path First

-P-

PCF	Point Coordinator Function
PDA	Personal Digital Assistant
PGP	Pretty Good Privacy
PKD	Public Key Distribution
PKI	Public Key Infrastructure
PKG	Private Key Generator

-Q-

QoS	Quality of Service

-R-

RAP	Rushing Attack Prevention
RERR	Route ERRor
RDP	Route Discovery Process
RFC	Request for Comments
RRDIR	Route Redirect
RREP	Route REPly message
RREQ	Route REQuest
REP	REPly
RSA	Rivest Shamir Adleman
RSP	Recorded Shortest Path
RTS-CTS	Request-To-Send - Clear-To-Send

-S-

SA	Security Association
SAODV	Secure AODV
SAR	Security-Aware ad hoc Routing
SEAD	Secure Efficient Ad hoc Distance-vector
SECTOR	SECure Tracking Of node encounteRs
SIM	Subscriber Identity Module
SLSP	Secure Link State routing Protocol
SMT	Secure Message Transmission
SN	Sequence Number
SPAAR	Secure Position Aided Ad hoc Routing
SPC	Shortest Path Confirmation
SRP	Secure Routing Protocol
SSL	Secure Sockets Layer
SUCV	Statistically Unique and Cryptographically Verifiable
SUNE	SUpervision in NEighborhood
SURO	SUpervision on ROute
SWAN	Secured Watchdog for Ad hoc Networks

-T-

TC	Topology Control
TCP	Transmission Control Protocol
TCSec	Securing Topology Control
TESLA	Timed Efficient Stream Loss-tolerant Authentication
TIK	TESLA with Instant Key disclosure
TDMA/TDD	Time Division Multiple Access/Time Division Duplex
TRP	Trust-based secure Routing Protocol
TRPS	TRP with SWAN
TD-SCDMA	Time Division-Synchronous Code Division Multiple Access
TSF	Time Synchronization Function
TTL	Time to live

-U-

UDP	User Datagram Protocol
UM-OLSR	University of Murcia - OLSR
UMTS	Universal Mobile Telecommunications System
UTRAN	UMTS Terrestrial Radio Access Network

-W-

WLAN	Wireless Local Area Networks
Wi-Fi	Wireless Fidelity
WiMAX	Worldwide Interoperability for Microwave Access
WPAN	Wireless Personal Area Networks

-Z-

ZRP	Zone Routing Protocol