THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PARIS 6 – PIERRE ET MARIE CURIE

discussed by

Daniele Raffo

on September 15, 2005

to obtain the degree of

Docteur de l'Université Paris 6

Discipline: Computer Science

Host laboratory: INRIA Rocquencourt

Security Schemes for the OLSR Protocol for Ad Hoc Networks

Thesis Director: Dr. Paul Mühlethaler

Jury

Reviewers:	Dr. Ana Cavalli	Institut National des Télécommunications
	Dr. Ahmed Serhrouchni	Ecole Nationale Supérieure des Télécommunications
Examiners:	Dr. François Baccelli	Ecole Normale Supérieure
	Dr. François Morain	Ecole Polytechnique
	Dr. Paul Mühlethaler	INRIA Rocquencourt
	Dr. Guy Pujolle	Université Paris 6
Guests:	Dr. Daniel Augot	INRIA Rocquencourt
	Dr. Philippe Jacquet	INRIA Rocquencourt

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PARIS 6 – PIERRE ET MARIE CURIE

présentée par

Daniele Raffo

le 15 Septembre 2005

pour obtenir le grade de

Docteur de l'Université Paris 6

Spécialité: Informatique

Laboratoire d'accueil: INRIA Rocquencourt

Schémas de sécurité pour le protocole OLSR pour les réseaux ad hoc

Directeur de Thèse: M. Paul Mühlethaler

Jury

Rapporteurs:	Mme Ana Cavalli	Institut National des Télécommunications
	M. Ahmed Serhrouchni	Ecole Nationale Supérieure des Télécommunications
Examinateurs:	M. François Baccelli	Ecole Normale Supérieure
	M. François Morain	Ecole Polytechnique
	M. Paul Mühlethaler	INRIA Rocquencourt
	M. Guy Pujolle	Université Paris 6
Invités:	M. Daniel Augot	INRIA Rocquencourt
	M. Philippe Jacquet	INRIA Rocquencourt

Dedicated to the memory of my grandfather Vincenzo

Abstract

Within the domain of wireless computer networks, this thesis examines the security issues related to protection of packet routing in ad hoc networks (MANETs). This thesis classifies the different possible attacks and examines in detail the case of OLSR (Optimized Link State Routing protocol). We propose a security architecture based on adding a digital signature, as well as more advanced techniques such as: reuse of previous topology information to validate the actual link state, cross-check of advertised routing control data with the node's geographical position, and intra-network misbehavior detection and elimination via flow coherence control or passive listening. Countermeasures in case of compromised routers are also presented. This thesis also assesses the practical problems concerning the choice of a suitable symmetric or asymmetric cipher, the alternatives for the algorithm of cryptographic keys distribution, and the selection of a method for signature timestamping.

Keywords

Ad hoc network, routing, link state, OLSR, security, digital signature

Résumé

Cette thèse examine les problématiques de sécurité liées à la protection du routage dans les réseaux ad hoc (MANETs). La thèse classifie les différentes attaques qui peuvent être portées et examine en détail le cas du protocole OLSR (Optimized Link State Routing). Une architecture de sécurisation basée sur l'ajout d'une signature numérique est étudiée et proposée. D'autres contre-mesures plus élaborées sont également présentées. Ces dernières incluent: la réutilisation d'informations topologiques précédentes pour valider l'état de lien actuel, l'évaluation de la véridicité des messages par analyse croisée avec la position géographique d'un noeud, et la détection des comportements suspects à l'intérieur du réseau par le contrôle de cohérence des flux ou l'écoute passif. La thèse analyse aussi les problèmes pratiques liées à la choix de l'algorithme de signature et la distributions des clés cryptographiques, et propose aussi des parades même en présence de noeuds compromis.

Mots clés

Réseau ad hoc, routage, état de lien, OLSR, sécurité, signature numérique

Contents

Foreword

My work examines the security issues related to the protection of the routing protocol in ad hoc networks, and more specifically of the OLSR protocol. OLSR has been developed by the HIPERCOM project group[1] at INRIA, the National Research Institute in Computer Science and Control, based in Rocquencourt, France.

OLSR was not designed with security in mind. Consequently, it is easy to find ways to maliciously perturb the correct functioning of the protocol. The aim of my doctoral researches, carried out in the HIPERCOM workgroup, was to explore the possible attacks and countermeasures to secure OLSR. This has led to the design of security extensions for OLSR, described in five papers published in international conferences [2, 130, 131, 132, 4] and in an INRIA Research Report [3]. I have also contributed in the writing of an Internet-Draft [30].

Structure of the thesis

Chapter 1 introduces the domain of wireless networking discussing the different types of architectures, and introduces the ad hoc networks by giving examples of routing protocols and a detailed overview of OLSR.

Chapter 2 handles the problem of system security, explaining the basics of cryptography. Chapter 3 provides a taxonomy of the attacks at the routing level in MANETs, and more specifically of the attacks against the OLSR protocol.

Chapter 4 outlines the countermeasures that can be taken in order to secure a wireless network, and gives some basic mechanisms (relying mainly on digests and digital signatures) to protect different routing protocols. A basic mechanism designed to secure the OLSR protocol is expounded in Chapter 5.

Chapter 6 debates the major choices that must be done in order to select a suitable cryptographic architecture, and discusses problematics related to the implementation of a Public Key Infrastructure on an ad hoc network,

[1] `http://hipercom.inria.fr/olsr`

with a proposal for OLSR. Chapter 7 offers a detailed view over the problem of a correct timestamping.

Chapter 8 introduces the topic of more advanced techniques to secure the routing protocol, in particular when the network has been compromised from the inside. The subsequent chapters present different studies concerning elaborated protection techniques for OLSR. Chapter 9 examines the insertion of old topology information in control messages to validate the actual link state, and Chapter 10 examines the use of GPS devices to cross-check advertised routing control data with information regarding the node's geographical position. Another detection technique, presented in Chapter 11, consists in the detection of intra-network misbehaviors; this is done by passive listening or controls on flow coherence. Last, Chapter 12 concludes the thesis.

Appendix A is an extended résumé of the thesis in the French language; every chapter of the thesis is condensed into a section of the résumé.

Style conventions

This thesis utilizes the following style conventions:

$A, B, C, \ldots$	nodes
t_0	time at instant 0
T_A	timestamp generated by A
$T_A(t_0)$	timestamp generated by A at instant 0
$x \leftarrow 0$	store the value 0 in x
$A \rightarrow B : \{M\}_A$	A sends the message M, signed by A, to B
$\langle X, p_X, T_X \rangle$	tuple
HELLO	OLSR (or derived protocol) control message
`Originator Address`	field of an OLSR message or packet

Acknowledgements

This doctoral thesis has been completed also thanks to many persons which contributed with suggestions, thoughts, and constructive criticisms. I take therefore the occasion to briefly mention them here.

I am greatly indebted to my thesis director Paul Mühlethaler, and with research director Philippe Jacquet, who welcomed me in the HIPERCOM project at INRIA. I am glad having spent my doctoral work within such a team. Paul guided me during my researches, and has been a very available and patient supervisor; his professional knowledge and constant support helped me proceed throughout my studies. I am grateful also to Guy Pujolle for accepting to be my thesis director at UPMC. Thanks to the INRIA for

the financial grant.

My thanks to all members of the jury of the thesis dissertation: François Baccelli, Ana Cavalli, François Morain, Paul Mühlethaler, Guy Pujolle, and Ahmed Serhrouchni. Besides participating in the jury, Ana Cavalli and Ahmed Serhrouchni accepted to devote their time in reviewing my thesis, providing very constructive comments and criticisms. I express my gratitude to François Baccelli, as well as to Mesaac Makpangou, also for being my pre-reviewers.

The whole INRIA HIPERCOM team deserves a special appreciation for an exceptionally friendly environment. In particular, I cannot certainly forget Thomas Clausen, who always provided me with his extremely useful and encouraging advices, and illustrated me the "1.3-year Ph.D. panic schedule". Thanks to Cédric Adjih and Géraud Allard for their useful ideas and for helping me in hacking my Linux box. Thanks to Pascale Minet for re-reading parts of the thesis. Thanks to Dang-Quan Nguyen, Amina Meraihi Naimi, Saadi Boudjit, and Adokoé Plakoo for their cooperation and their valuable tips.

Thanks very much to Daniel Augot and Raghav Bhaskar (INRIA CODES) and, again, to François Morain (LIX) for the helpful discussions on cryptography, in spite of their busy timetable. Thanks to Xiaoyun Xue (ENST) for spotting a flaw in the ADVSIG architecture. Joe Macker (NRL) and his group, Justin Dean included, Andreas Hafslund and Eli Winjum (UniK), and Ricardo Staciarini Puttini (UNB) contributed with discussions and links about securing OLSR.

Richard James and Ishak Binudin helped in correcting the manuscript; thanks to Richard also for being always available to examine my scientific papers.

Several people helped me in a way or another during these three years. Therefore I take the occasion to thank, in no particular order, Marco Perisi, Marfi Giagu with Patrick Marcellin, Xanthi Kapsosideri, Eufrosine Andreou, Anne Dautzenberg, Cécile Bredelet, Charles Saada, Karina with Erik Fjeldstad, Jacques Henry, Claire Alexandre, Eliane Launay with Gilles Scagnelli, Aïssa Amoura, Christian Tourniaire, Danielle Croisy, Saholy with Stéphane Grolleau, and Vincent Lucquiaud.

Thanks to Matteo, Salvio, Federico, Marta, and all others for our Italians-online community in Paris!

My deepest thanks, and apologies, to Sophie for her support, patience and understanding during the writing of my thesis.

Last but not least, thanks a lot to my family, for always supporting me during my studies abroad.

Chapter 1

Introduction to wireless networking

In wireless networks [102, 45], computers are connected and communicate with each other not by a visible medium, but by emissions of electromagnetic energy in the air.

The most widely used transmission support is radio waves. Wireless transmissions utilize the microwave spectre: the available frequencies are situated around the 2.4 GHz ISM (Industrial, Scientific and Medical) band for a bandwidth of about 83 MHz, and around the 5 GHz U-NII (Unlicensed-National Information Infrastructure) band for a bandwidth of about 300 MHz divided into two parts. The exact frequency allocations are set by laws in the different countries; the same laws also regulate the maximum allotted transmission power and location (indoor, outdoor). Such a wireless radio network has a range of about 10–100 meters to 10 Km per machine, depending on the emission power, the data rate, the frequency, and the type of antenna used. Many different models of antenna can be employed: omnis (omnidirectional antennas), sector antennas (directional antennas), yagis, parabolic dishes, or waveguides (cantennas).

The other type of transmission support is the infrared. Infrared rays cannot penetrate opaque materials and have a smaller range of about 10 meters. For these reasons, infrared technology is mostly used for small devices in WPANs (Wireless Personal Area Networks), for instance to connect a PDA to a laptop inside a room.

1.1 Standards

There are presently three main standards for wireless networks: the IEEE 802.11 family, HiperLAN, and Bluetooth.

1.1.1 IEEE 802.11

IEEE 802.11 [108] is a standard issued by the IEEE (Institute of Electrical and Electronics Engineers). From the point of view of the physical layer, it defines three non-interoperable techniques: IEEE 802.11 FHSS (Frequency Hopping Spread Spectrum) and IEEE 802.11 DSSS (Direct Sequence Spread Spectrum), which use both the radio medium at 2.4 GHz, and IEEE 802.11 IR (InfraRed). The achieved data rate is 1–2 Mbps. This specification has given birth to a family of other standards:

IEEE 802.11a [71] (marketed as Wi-Fi5) operates in the 5 GHz U-NII band using the OFDM (Orthogonal Frequency Division Multiplexing) transmission technique, and has a maximum data rate of 54 Mbps. IEEE 802.11a is incompatible with 802.11b, because they use different frequencies.

IEEE 802.11b [72] (marketed as Wi-Fi) is the de facto standard in wireless networking, and operates in the 2.4 GHz ISM band. The data rate is 1, 2, 5 or 11 Mbps, automatically adjusted depending on signal strength. The transmission range depends on the data rate, varying from 50 meters indoor (200 meters outdoor) for 11 Mbps, to 150 meters indoor (500 meters outdoor) for 1 Mbps; the transmission range is also proportional to the signal power.

IEEE 802.11g [73] operates in the 2.4 GHz band and has a data rate of up to 20 Mbps. It uses both OFDM and DSSS to ensure compatibility with the IEEE 802.11b standard.

Another standard currently under development, IEEE 802.16 [75] (marketed as WiMAX), is designed for WMANs (Wireless Metropolitan Area Networks) and therefore to overcome the range limitations of IEEE 802.11. It operates on frequencies from 10 to 66 GHz, and should ensure network coverage for several square Km. From the IEEE 802.16 standard derives IEEE 802.16a, that operates on the 2-11 GHz band and should solve the line-of-sight problems deriving from using the 10-66 GHz band.

Channel access techniques

The crucial point in channel access techniques for wireless networks is that it is not possible to transmit and to sense the carrier for packet collisions at the same time. Therefore there is no way to implement a CSMA/CD (Carrier Sense Multiple Access / Collision Detection) protocol such as in the wired Ethernet.

IEEE 802.11 uses a channel access technique of type CSMA/CA, which is meant to perform Collision Avoidance (or at least to try to). The CSMA/CA protocol states that a node, upon sensing that the channel is busy, must

wait for an interframe spacing before attempting to transmit, then choose a random delay depending on the Contention Window.

The reception of a packet is acknowledged by the receiver to the sender. If the sender does not receive the acknowledgement packet, it waits for a delay according to the binary exponential backoff algorithm, which states that the Contention Window size is doubled at each failed try.

Unicast data packets are sent using a more reliable mechanism. The source transmits a RTS (Request To Send) packet for the destination, which replies with a CTS (Clear To Send) packet upon reception. If the source correctly receives the CTS, it sends the data packet.

1.1.2 HiperLAN

HiperLAN (High Performance Radio LAN) is a standard issued by the ETSI (European Telecommunications Standard Institute), and a competitor of IEEE 802.11. It defines two kinds of networks:

HiperLAN 1 [42] uses the 5 GHz band and offers a data rate of 10–20 Mbps.

HiperLAN 2 [44, 43] uses the 5 GHz band and offers a data rate up to 54 Mbps.

A related standard is HiperMAN, rival of IEEE 802.16 and aimed at providing metropolitan area coverage. It operates in the 2–11 GHz band.

1.1.3 Bluetooth

Bluetooth[1] is a standard designed by a consortium of private companies such as Agere, Ericsson, IBM, Intel, Microsoft, Motorola, Nokia and Toshiba. Bluetooth operates in the 2.4 GHz band using FHSS and has a short range of action of about 10 meters. For such characteristics and its low cost, Bluetooth is fit for small WPANs and is also employed to connect peripherals such as keyboards, printers, or mobile phone headsets. Bluetooth radio technology works in a master-slave fashion, and each device can operate as master or as slave. Communications are organized in small networks called *piconets*, each piconet being composed of a master and 1–7 active slaves. Multiple piconets can overlap to form a *scatternet*.

1.2 Architecture

A wireless network can be structured to function in either BSS (Basic Service Set) or IBSS (Independent Basic Service Set) mode. The two modes affect the topology and the mobility capabilities of the machines (*nodes*) that compose the network.

[1] `http://www.bluetooth.org`

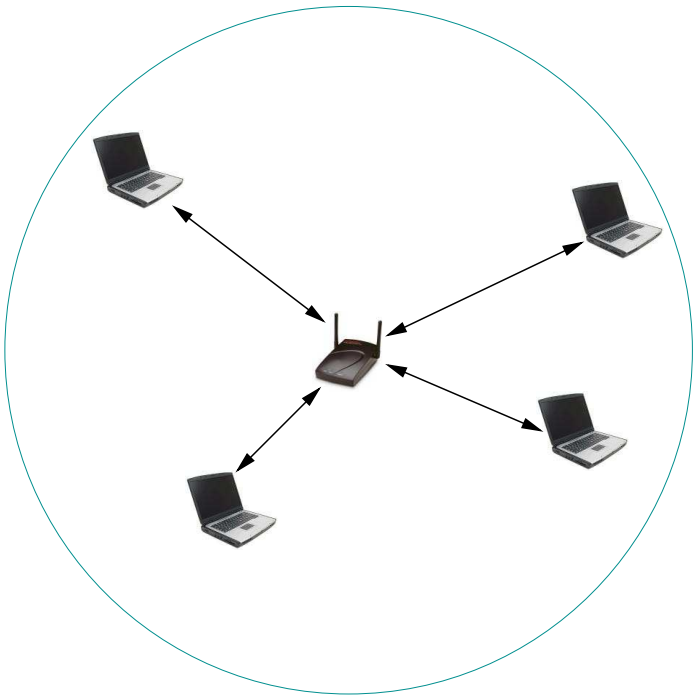

Figure 1.1: BSS mode: an Access Point and its network cell.

1.2.1 BSS mode

In BSS mode, also called *infrastructure mode*, a number of mobile nodes are wirelessly connected to a non-mobile Access Point (AP), as in Figure 1.1. Nodes communicate via the AP, which may also provide connectivity with an external wired network e.g. the Internet. Several BSS networks may be joined to form an ESS (Extended Service Set).

1.2.2 IBSS mode

The IBSS mode, also called *peer to peer* or *ad hoc mode*, allows nodes to communicate directly (point-to-point) without the need for an AP, as in Figure 1.2. There is no fixed infrastructure. Nodes need to be in range with each other in order to communicate.

1.2.3 Ad hoc network

An *ad hoc network*, or MANET (Mobile Ad hoc NETwork), is a network composed only of nodes, with no Access Point. Messages are exchanged and relayed between nodes. In fact, an ad hoc network has the capability of making communications possible even between two nodes that are not

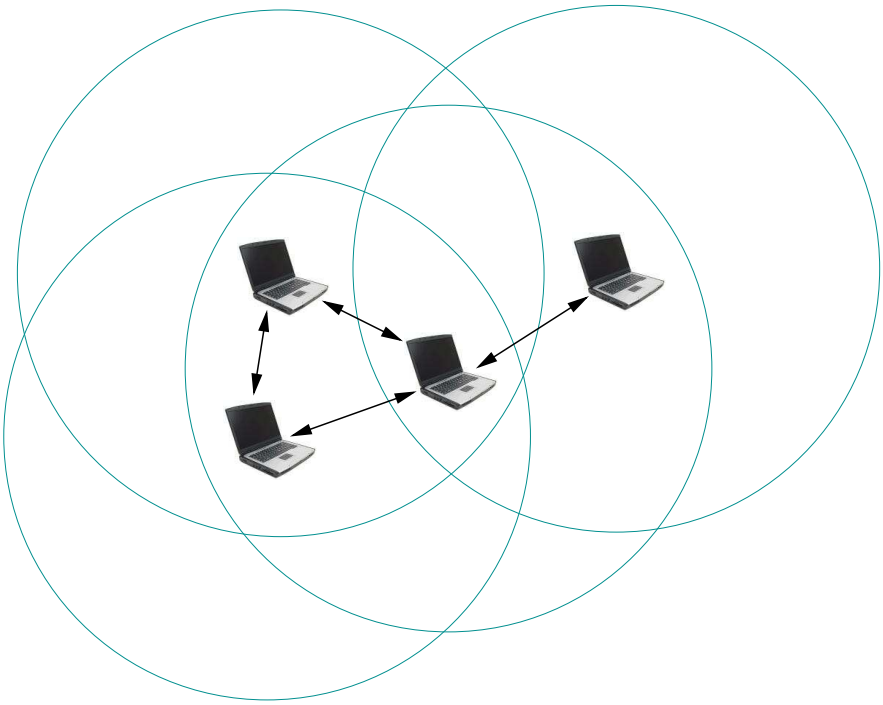

Figure 1.2: IBSS mode.

in direct range with each other: packets to be exchanged between these two nodes are forwarded by intermediate nodes, using a routing algorithm.[2] Hence, a MANET may spread over a larger distance, provided that its ends are interconnected by a chain of links between nodes (also called *routers* in this architecture). In the ad hoc network shown in Figure 1.3, node A can communicate with node D via nodes B and C, and vice versa.

A *sensor network* is a special class of ad hoc network, composed of devices equipped with sensors to monitor temperature, sound, or any other environmental condition. These devices are usually deployed in large number and have limited resources in terms of battery energy, bandwidth, memory, and computational power.

1.3 Advantages and disadvantages

A wireless network offers important advantages with respect to its wired homologue:

- The main advantage is that a wireless network allows the machines to

[2] An ad hoc network must not be confused with a network in ad hoc mode. In ad hoc mode, nodes do not relay packets (multihop not implemented).

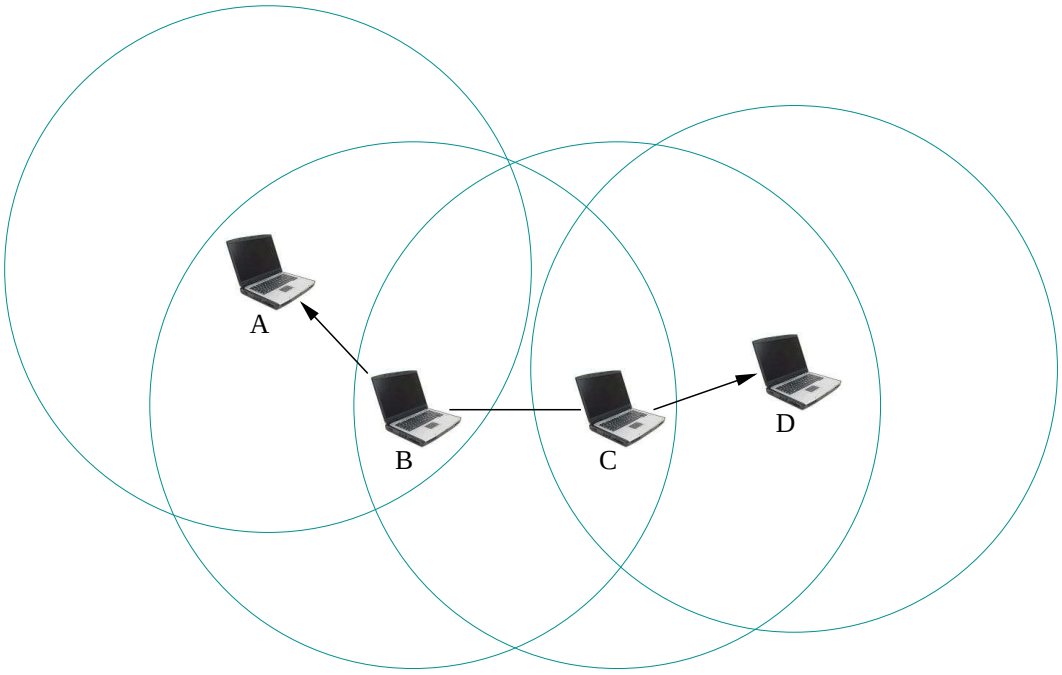

Figure 1.3: An ad hoc network.

be fully mobile, as long as they remain in radio range.

- Even when the machines do not necessarily need to be mobile, a wireless network avoids the burden of having cables between the machines. From this point of view, setting a wireless network is simpler and faster. In several cases, because of the nature and topology of the landscape, it is not possible or desirable to deploy cables: battlefields, search-and-rescue operations, or standard communication needs in ancient buildings, museums, public exhibitions, train stations, or inter-building areas.

- While the immediate cost of a small wireless network (the cost of the network cards) may be higher than the cost of a wired one, extending the network is cheaper. As there are no wires, there is no cost for material, installation and maintenance. Moreover, mutating the topology of a wireless network – to add, remove or displace a machine – is easy.

On the other hand, there are some drawbacks that need to be pondered:

- The strength of the radio signal weakens (with the square of the distance), hence the machines have a limited radio range and a restricted

scope of the network. This causes the well-known *hidden station* problem [149]: consider three machines A, B and C, where both A and C are in radio range of B but they are not in radio range of each other. This may happen because the $A-C$ distance is greater than the $A-B$ and $B-C$ distances, as in Figure 1.4, or because of an obstacle between A and C. The hidden station problem occurs whenever C is transmitting: when A wants to send to B, A cannot hear that B is busy and that a message collision would occur, hence A transmits when it should not; and when B wants to send to A, it mistakenly thinks that the transmission will fail, hence B abstains from transmitting when it would not need to.

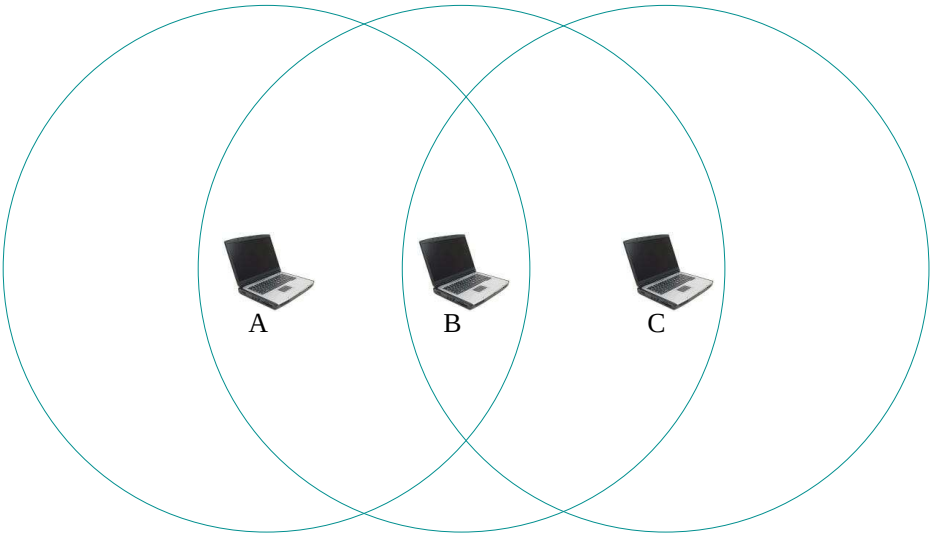

Figure 1.4: The hidden station problem.

- The site variably influences the functioning of the network: radio waves are absorbed by some objects (brick walls, trees, earth, human bodies) and reflected by others (fences, pipes, other metallic objects, water). Wireless networks are also subject to interferences by other equipment that shares the same band, such as microwave ovens and other wireless networks.
- Considering the limited range and possible interferences, the data rate is often lower than that of a wired network. However, nowadays some standards offer data rates comparable to those of Ethernet.
- Due to limitations of the medium, it is not possible to transmit and to listen at the same time, therefore there are higher chances of message collisions. Collisions and interferences make message losses more likely.

- Being mobile computers, the machines have limited battery and computation power. This may entail high communication latency: machines may be off most of the time (*doze state* i.e. power-saving mode) and turning on their receivers periodically, therefore it is necessary to wait until they wake up and are ready to communicate.

- As data is transmitted over Hertzian waves, wireless networks are inherently less secure (see Chapter 3). In fact, transmissions between two computers can be eavesdropped by any similar equipment that happens to be in radio range.

1.4 Routing protocols for ad hoc networks

In ad hoc networks, to ensure the delivery of a packet from sender to destination, each node must run a routing protocol and maintain its routing tables in memory.

Routing protocols can be classified into the following categories: reactive, proactive, and hybrid. There exists nowadays almost one hundred routing protocols, many standardized by the IETF (Internet Engineering Task Force) and others still at the stage of Internet-Draft. This section gives, for each category, an overview of the most important ones.

1.4.1 Reactive protocols

Under a *reactive* (also called *on-demand*) protocol, topology data is given only when needed. Whenever a node wants to know the route to a destination node, it floods the network with a route request message. This gives a reduced average control traffic, with bursts of messages when packets need being routed, and an additional delay due to the fact that the route is not immediately available.

- DSR (Dynamic Source Routing) [83, 82] uses a *source routing* mechanism, i.e. the complete route for the packet is included in the packet header. This avoids path loops. To discover a route, a node floods a Route Request and awaits the answers; any receiving node adds its address to the Route Request and retransmits the packet. Once the packet has reached its final destination node, the latter reverses the route and sends the Route Reply packet. This is possible if the MAC protocol permits bidirectional communications; otherwise, the destination node performs another route discovery back to the originator. Every node maintains also a route cache, which avoids doing a route discovery for already known routes. A mechanism of route maintenance allows the originator node to be alerted about link breaks in the route.

- AODV (Ad hoc On-demand Distance Vector routing) [119, 121] is a *distance vector* routing protocol, i.e. routes are advertised as a vector of direction and distance. To avoid the Bellman-Ford "counting to infinity" problem and routing loops, sequence numbers are utilized for control messages. To find a route to a destination, a node broadcasts a RREQ (Route REQuest) message. The RREQ is relayed by receiving nodes until it reaches the destination or an intermediate node with a fresh route (i.e. a route with an associated sequence number equal or greater than that of the RREQ) to destination. Afterward, a RREP (Route REPly) message is unicast by the destination to the originator of the RREQ. RERR (Route ERRor) messages are used to notify nodes about link breaks.

- DSDV (Destination-Sequenced Distance-Vector routing) [120] is another distance vector routing protocol, which requires each node to advertise its routing table to its neighbors. Route information contains a route sequence number, the destination's address, the destination's distance in hops, and the sequence number of the information received regarding the destination as stamped by the destination itself.

1.4.2 Proactive protocols

In opposition, *proactive* (also called *periodic* or *table driven*) protocols are characterized by periodic exchange of topology control messages. Nodes periodically update their routing tables. Therefore, control traffic is more dense but constant, and routes are instantly available.

- OLSR (Optimized Link State Routing) is a link state routing protocol, described in detail in Section 1.4.4.

- OSPF (Open Shortest Path First) [110, 32] is another link state routing protocol, issued from the very first link state protocols used in the ARPANET packet switching network. OSPF maintains information about network topology in a database stored in every node. From this database, every node builds a shortest-path tree to route a packet to its destination. Neighbor discovery is accomplished through exchange of HELLO packets.

- FSR (Fisheye State Routing) [54, 118] is a scalability-supporting link state protocol. Each node broadcasts link state information of a destination to its neighbors, with a frequency inversely proportional to the destination's distance in hops; i.e. information about distant nodes is broadcast less often. Therefore, every node has a precise knowledge of its local neighborhood while knowledge of distant nodes is less precise

(hence the name "Fisheye"). This makes the routing of a packet accurate near the source and the destination. FSR is proficient in handling large networks.

- TBRPF (Topology dissemination Based on Reverse-Path Forwarding) [115] is a link state protocol in which each node builds a source tree using partial topology information stored in its topology table. The tree provides paths to all reachable nodes and is computed using a modified Dijkstra algorithm. Each node periodically shares part of its tree with its neighbors. Differential HELLO messages, which report only changes in neighbors' status, are used for neighbor discovery.

- ADV (Adaptive Distance Vector routing) [18] is a proactive protocol, but with some reactive characteristics. Each node shares its route information with its neighbors, according to the Distributed Bellman-Ford distance vector algorithm. However, in ADV a node maintains only routes to nodes that are currently receivers of any active connection. Furthermore, the frequency of route updates varies depending on the load and mobility of the network. ADV therefore quickly adapts itself to sudden changes on the network load.

- STAR (Source Tree Adaptive Routing) [49] uses a source tree, computed by every node, in order to route packets. Every node then shares its whole tree with its neighbors.

- LANMAR (LANdMARk routing) [52, 53] is a routing protocol aimed at large networks divided into logical groups. It assumes that every node is identified by an addressing scheme containing the group ID and host ID. Nodes use a scoped routing protocol, e.g. FSR, to learn routes to nearby nodes. Every group elects a landmark; packets are routed towards the landmark corresponding to the group ID of the destination, then delivered directly to the destination.

- WRP (Wireless Routing Protocol) [111] is based on a path-finding algorithm that reduces the probability or routing loops. In WRP, each node shares its routing tables with its neighbors, by communicating the distance and second-to-last hop to each destination. Nodes send an acknowledgement upon reception of update routes. Each nodes maintain a distance table, a routing table, a link-cost table, and a message retransmission list.

- WIRP (Wireless Internet Routing Protocol) [48] is a routing protocol designed to operate with Wireless Internet Gateways (WINGs), improved self-adapting routers for the wireless ad hoc environment. The radio device is controlled by the FAMA-NCS protocol, which eliminates the hidden station problem in single-channel networks. WIRP

interoperates with FAMA-NCS for the link sensing mechanism. Each node builds a hierarchical routing tree and distributes it incrementally to its neighbors, by communicating only the distance and the second-last-hop to each destination. Route updates must be acknowledged by each node.

1.4.3 Hybrid protocols

Hybrid protocols have both the reactive and proactive nature. Usually, the network is divided into regions, and a node employs a proactive protocol for routing inside its near neighborhood's region and a reactive protocol for routing outside this region.

- ZRP (Zone Routing Protocol) [57] defines for every node a radius (in number of hops) inside which packets are routed using a proactive routing protocol. Routes for nodes outside the radius are discovered using a reactive routing protocol. The working mode of ZRP is specified locally by IARP (IntrAzone Routing Protocol) [59], and for the rest of the network (outside the radius) by IERP (IntErzone Routing Protocol) [58].
- CBRP (Cluster Based Routing Protocol) [81] divides the network into overlapping or disjoint node clusters, each cluster being 2 hops in diameter. For every cluster, the cluster head node has the duty of exchanging route discovery messages with other cluster heads. A proactive routing protocol is used inside every cluster, while inter-cluster routes are discovered reactively via route requests.

1.4.4 The Optimized Link State Routing protocol

The Optimized Link State Routing (OLSR) protocol [31, 79, 29] is a proactive link state routing protocol for ad hoc networks.

The core optimization of OLSR is the flooding mechanism for distributing link state information, which is broadcast in the network by selected nodes called *Multipoint Relays (MPR)*. As a further optimization, only partial link state is diffused in the network. OLSR provides optimal routes (in terms of number of hops) and is particularly suitable for large and dense networks.

Specifications of the protocol were first described in an Internet-Draft in February 2000, and were finalized in RFC 3626 [31] in October 2003; there is also a draft for the version 2 of the protocol [27]. Several implementations exist at this day: OOLSR (the original, object-oriented implementation of OLSR by INRIA HIPERCOM), nlrolsrd (by the U.S. Naval Research Laboratory), OLSR_Niigata (by Niigata University), Qolyester (a Quality-of-Service enhanced version by LRI), OLSR11win (by the GRC, Universitat Politècnica de València), the olsr.org OLSR daemon (by UniK, University of

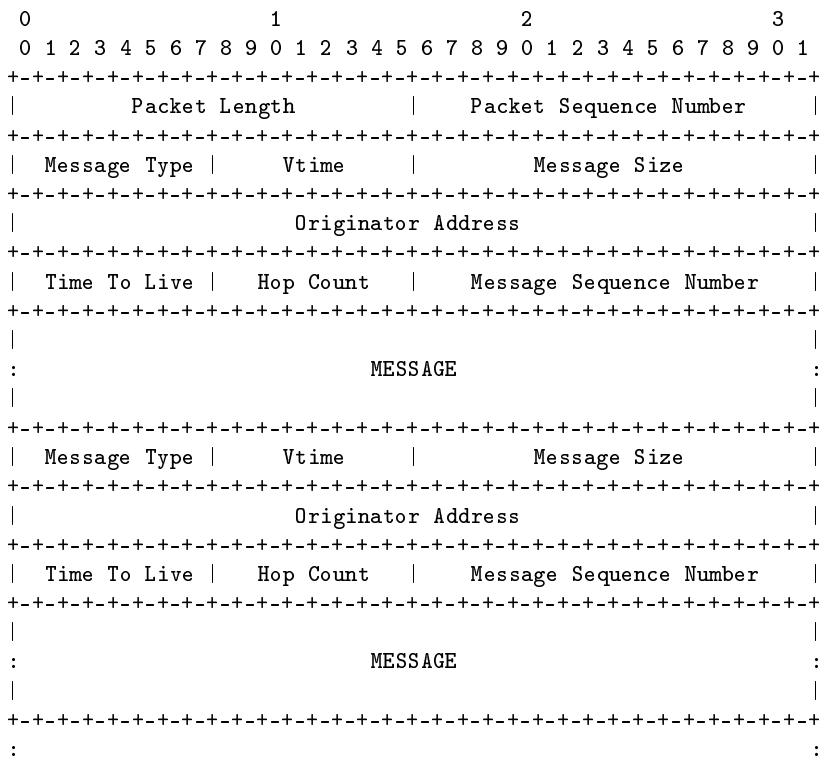

Figure 1.5: OLSR packet format.

Oslo), H-OLSR (by Hitachi, Ltd.), and CRC OLSR (by the Communication Research Centre in Canada). A multicast extension [95] has been proposed and is the object of an Internet-Draft (MOLSR) [80].

OLSR message and packet format

OLSR control messages are communicated using a transport protocol defined by a general packet format, given in Figure 1.5. Each packet encapsulates several control messages into one transmission.

Control traffic in OLSR is exchanged through two different types of messages: HELLO and TC (Topology Control) messages. HELLO messages, shown in Figure 1.6, are exchanged periodically among neighbor nodes, in order to detect links to neighbors and to signal MPR selection. TC messages, shown in Figure 1.7, are periodically flooded to the entire network, in order to diffuse link state information to all nodes.

The other OLSR control messages are MID (Multiple Interface Declaration) and HNA (Host and Network Association). MID and HNA messages are emitted only by nodes that have multiple interfaces. To avoid collisions, the OLSR protocol adds an amount of jitter to the interval at which all control

messages are generated.

While messages may potentially be broadcast to the entire network, packets are transmitted only between neighbor nodes. The unit of information subject to being forwarded is a "message". An individual OLSR control message can be uniquely identified by its `Originator Address` and `Message Sequence Number` (MSN), both from the message header. The `Originator Address` field specifies the originator of a message, and does not change as the message is relayed around the network; the address contained in this field is different (except at the first hop, when the message is created) from the IP header source address, which is changed at each hop to the address of the retransmitting node.

A node may receive the same message several times. Therefore, to avoid processing and sending multiple times the same message, a node records information about each received message. This information is stored in a tuple consisting of the message's originator address, the MSN, a boolean value indicating whether the message has already been retransmitted, the list of interfaces on which the message has been received, and the tuple's expiration time. All tuples are maintained in the Duplicate Set (also known as Duplicate Table) of the node.

The common packet format allows individual messages to be piggybacked and transmitted together in one emission, if allowed by the MTU size. Therefore different kind of control messages can be emitted together, although processed and forwarded differently in each node; e.g. HELLO messages are not forwarded while all other control messages are.

OLSR does not handle unicast communications: a message from a node is either transmitted to all its neighbors or to all nodes in the network.

HELLO messages contain a list of neighbors from which control traffic has been heard (but with which bidirectional communication is not yet confirmed), a list of neighbors with which bidirectional communication has been established, and a list of neighbors that have been selected to act as a Multipoint Relay for the originator of the HELLO message. Each `Neighbor Interface Address` field contains the address of an advertised neighbor, and the relevant `Link Code` field contains its link status as a combination of Link Type and Neighbor Type. Table 1.1 lists the constants' values for this last field, as specified by the protocol documentation [31].

Upon receiving a HELLO message, a node examines the lists of addresses. If its own address is included in the addresses encoded in the HELLO message, bidirectional communication is possible (symmetrical link) between the originator and the recipient of the HELLO message, i.e. the node itself.

In addition to information about neighbor nodes, periodic exchange

Link Types	
UNSPEC_LINK	No information
ASYM_LINK	Link is asymmetrical, i.e. neighbor is heard
SYM_LINK	Link is symmetrical
LOST_LINK	Link has been lost
Neighbor Types	
SYM_NEIGH	Neighbor is symmetric
MPR_NEIGH	Neighbor has been selected as MPR
NOT_NEIGH	Node is no longer / not yet symmetric neighbor

Table 1.1: Constants for the `Link Code` field in a HELLO.

of HELLO messages allows each node to maintain information describing the links between neighbor nodes and nodes which are two hops away. This information is recorded in a nodes 2-hop neighbor set and is utilized for MPR optimization.

HELLO messages are exchanged periodically between neighbor nodes only, and are not forwarded further.

TC messages have the purpose to diffuse link state information, and more precisely information about the "last hop", to the entire network. A TC message contains a set of symmetric neighbors (i.e. neighbors which have at least one symmetrical link with the originator of the TC message) [28], each one contained in a `Advertised Neighbor Main Address` field. TC messages are periodically flooded to the entire network, exploiting the MPR optimization. Only nodes which have been selected as an MPR generate (and relay) TC messages.

The TC message bears an `ANSN` field which contains the Advertised Neighbor Sequence Number. This number is associated with the node's advertised neighbor set, and is incremented each time the node detects a change in this set.

MID messages are emitted only by a node with multiple OLSR interfaces, in order to announce information about its interface configuration to the network. A MID message contains a list of addresses, each address belonging to an OLSR interface of the sending node.

HNA messages are emitted only by a node with multiple non-MANET interfaces, and have the purpose of providing connectivity from a OLSR network to a non-OLSR network. The gateway sends HNA messages containing a list of addresses of the associated networks and their netmasks.

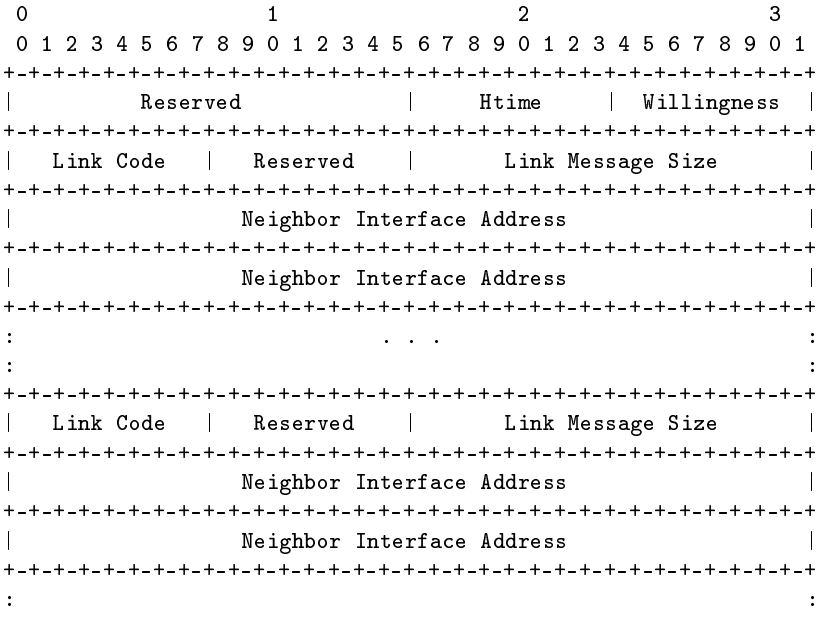

Figure 1.6: HELLO message format.

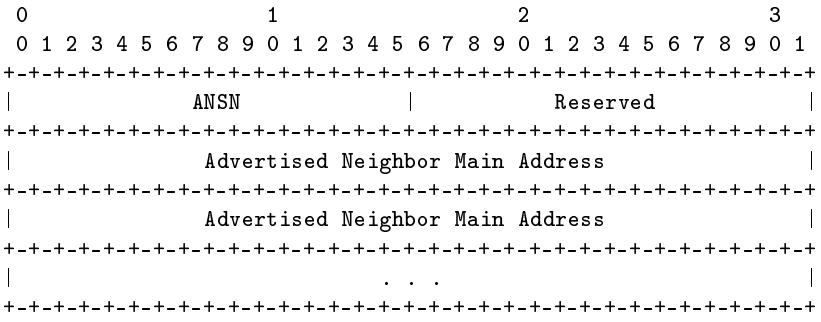

Figure 1.7: TC message format.

Multipoint Relay selection and signaling

The OLSR backbone for message flooding is composed of Multipoint Relays. Each node must select MPRs from among its symmetric neighbor nodes such that a message emitted by a node and repeated by the MPR nodes will be received by all nodes two hops away. In fact, in order to achieve a network-wide broadcast, a broadcast transmission needs only be repeated by just a subset of the neighbors: this subset is the MPR set of the node. Hence only MPR nodes relay TC, MID, and HNA messages.

Figure 1.8 shows the node in the center, with neighbors and 2-hop neighbors, broadcasting a message. In (a) all nodes retransmit the broadcast, while in (b) only the MPRs of the central node retransmit the broadcast.

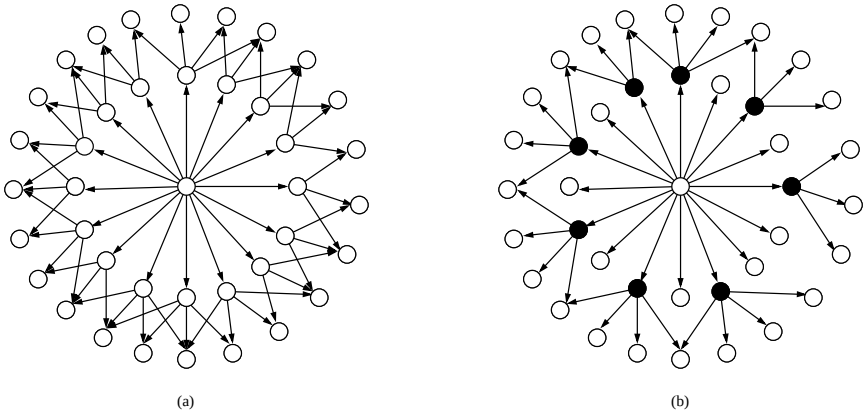

Figure 1.8: Pure flooding and MPR flooding.

The MPR set of a node is computed heuristically [129]. MPR selection is performed based on the 2-hop neighbor set received through the exchange of HELLO messages, and is signaled through the same mechanism. Each node maintains an *MPR selector set*, describing the set of nodes that have selected it as MPR.

Security considerations

The standard OLSR specification document does not take account of security measures. It enumerates possible vulnerabilities to which OLSR is subject. These vulnerabilities include breach of confidentiality, breach of integrity, non-relaying, replay, and interaction with an insecure external routing domain.

We give in Chapter 2 a brief overview on system security, and in Chapter 3 a detailed description of the attacks against OLSR and against the routing protocols in general. A mechanism designed to secure the OLSR protocol is presented in Chapter 5.

Chapter 2

System security

A *secure system* may be defined as a system that does exactly what its designers conceived it for and does not show any unexpected behavior, even when an attacker tries to make the system act differently.

A definition of *security* is indeed incomplete without specifying against who or what the system is secured. Furthermore, as absolute security is impossible to obtain, a report about the cost/benefit balance must be established.

It must be recalled that enforcing security requires that the defender covers all points of possible attack, as, for the attacker, it is sufficient to focus its efforts on one weak point in order to succeed. Therefore a system is only as secure as its less reliable security point. This is synthesized in the widely known expression: "a chain is as strong as its weakest link".

When talking about security of a communications network, there are different areas in which this topic applies. The major security goals are defined with the terms which follow; for each goal, the associated attack is identified. The name can describe either the functioning of the attack or its effect.

- ***Confidentiality, privacy, secrecy*** $\Longleftrightarrow$ ***Eavesdropping***
  Confidentiality means that the transmitted information is only disclosed to authorized parties. Sensitive information disclosed to an adversary could have severe consequences.

- ***Integrity*** $\Longleftrightarrow$ ***Message tampering***
  Integrity assumes that a message is not altered in transit between sender and receiver. Messages could be corrupted due to network malfunctioning or malicious attacks.

- ***Non-repudiation*** $\Longleftrightarrow$ ***Message forgery***
  Non-repudiation means that the originator of a message cannot deny having sent the message. An attacker could forge a wrong message that appears to be originating from an authorized party, with the aim

of making the party the culprit. If non-repudiation is guaranteed, the receiver of a wrong message can prove that the originator sent it, and that therefore the originator misbehaved.

Other security goals may be more difficult to achieve. Note that attacks can be combined, e.g. the intruder may break into the system in order to prepare a DoS from inside, or may perform eavesdropping with the purpose of later gaining unauthorized access.

- ***Authentication*** $\Longleftrightarrow$ ***Identity spoofing, impersonation***
  Authentication ensures the identity of the party with which communications are exchanged, before granting it access to the network. Without authentication, an attacker could masquerade as a legitimate party (*identity spoofing*) and interfere with the security of the network.

- ***Access control*** $\Longleftrightarrow$ ***Breaking, unauthorized access***
  Access control means that only authorized parties can participate in the communications; any other entity is denied access. Access control presumes authentication of the party trying to have access to the network.

- ***Service availability*** $\Longleftrightarrow$ ***Denial of Service***
  Service availability must guarantee that all resources of the communications network are always utilizable by authorized parties. An attacker may launch a *Denial of Service (DoS)* attack by saturating the medium, jamming the communications, or keeping the system resources busy in any other way. The aim here is just to impede authorized parties from having access to the resources, thereby making the network unusable.

Many security countermeasures are achieved by the use of cryptography [139, 13].

2.1 Cryptography basics

Encryption is the process of disguising a message in such a way that it hides its content; the operation consists in transforming the message from *plaintext* to *ciphertext*. The inverse process is called *decryption*.

It is also possible to add a *message digest*, also called a *hashing* or *digital fingerprint*, to the message so that the integrity of the message can be verified.

Signing a message means, instead, to add a sequence of bits (a *digital signature*) to the message in order to identify its real originator.

These techniques are performed by using a cryptographic algorithm (*cipher*) and a *key*, whose format depends on the algorithm used. It is often necessary to apply more than one technique, i.e. a message can be encrypted and then digitally signed.

With respect to the aforementioned security attributes:

- the encryption provides confidentiality, because the messages is transmitted in ciphertext, and only the owner of the key can decrypt the ciphertext;
- the message digest provides integrity;
- the signature provides non-repudiation, as only the owner of the key could have generated it.

Authentication, and subsequent access control, is more complicated to obtain and requires the use of more advanced cryptographic primitives, while service availability is not the concern of cryptography.

It is likely that information that was true at some time in the past may not be true anymore in the present. A common problem is that, even assuming a digest or signature is successfully checked, previously transmitted messages can be sent again by an attacker. That is, an intruder may record a bulk of messages and re-send them some time later; these messages, if they cannot be identified as old (by some definition of "old"), will be accepted as valid because they are properly signed. This is known as *replay attack*, and may easily disrupt communications. To oppose replay attacks, messages usually embed a piece of time information, called *timestamp*, describing the time at which the message was generated. The timestamp is included in the computation of the signature. Timestamps are discussed in detail in Chapter 7.

An adversary may exploit possible weaknesses in cryptographic functions. For instance, when relaying a control message with digest from one node to another, an attacker may replace the original message with a forged one which, due to a flaw in the digesting algorithm, has the same digital fingerprint. The adversary discovers these flaws using different techniques e.g. plaintext-chosen or brute-force attacks, depending on the data available to work on. These kinds of codebreaking attacks (*cryptanalysis*) are aimed against the cryptographic layer, and do not require the disclosure of any key to the attacker. However, when designing security schemes that rely on cryptography, it is usually assumed that cryptographic primitives are robust against these attacks.

Two branches of cryptography exist: symmetric cryptography and asymmetric cryptography. Each is useful to perform different functions.

2.1.1 Symmetric cryptography

Symmetric cryptography (also called *secret key cryptography*, *single key cryptography*, or *one key cryptography*) is the most ancient form of cryptography. Symmetric cryptography is based on symmetric key algorithms, i.e. algorithms where the encryption key and the decryption key are the same (or,

more broadly, where the encryption key can be computed from the decryption key and vice versa). The sender and the receiver of a message must agree on a secret shared key, which will henceforth be used to encrypt, decrypt, and generate a digest on exchanged messages.

Encryption

Some of the symmetric algorithms for encryption are: DES with its improvements Triple DES and AES, IDEA, LOKI, Lucifer, Skipjack, Vernam (also known as one-time pad), RC2, and RC4.

To this class of algorithms also belong the ancient substitution and transposition ciphers, like Caesar, Mary Stuart's, Pigpen, Vigenere, Playfair, and ADFGVX. These ciphers were in use centuries ago, in the pre-computer era, and are not used anymore because they are easy to break by applying cryptanalysis.

Message digest

Symmetric algorithms make large use of *hash functions* [106] for digesting. A hash function h maps a bitstring of arbitrary finite length to another bitstring of fixed length n, where n depends on h. The hash function hence outputs a *hash value* which is a condensed representative image of the bitstring fed in input. Changing just one bit of the input string results in a very different hash value in output; this is known as the *avalanche effect*.

A hash function h should have the following properties:

- be one-way, i.e. given an output y it is computationally infeasible to find an input x such that $h(x) = y$ (*preimage resistance*);
- given an input x it is computationally infeasible to find another input $x' \neq x$ such that $h(x') = h(x)$ (*second preimage resistance*);
- it is computationally infeasible to find two inputs x, x', with $x \neq x'$, such that $h(x) = h(x')$ (*collision resistance*).

Examples of hash functions are MD5 (Message Digest 5) [134] which is the successor of MD4, Snefru, RIPEMD-160, and the class of SHA (Secure Hash Algorithm) functions [113] such as SHA-1 [40] and SHA-256.

Cryptographic literature often references a *random oracle* [10, 23]. A random oracle is a theoretical model of a "perfect" hash function which returns an answer uniformly selected amongst all possible answers.

A hash function may be used in conjunction with a secret shared key (e.g. by concatenating the key to the hash input) to construct a keyed hash function. In this case, the digest is more often called *Message Authentication*

Code (MAC)[1]. This is the foundation of the HMAC mechanism [9, 91]. The resulting keyed hash function is called with a name that depends on the hash function used, for instance HMAC-MD5, HMAC-RIPEMD, or HMAC-SHA1.

2.1.2 Asymmetric cryptography

In *asymmetric cryptography* (also called *public key cryptography*), there is a key for encryption (*public key*) and another key for decryption (*private key* or *secret key*). A public and its companion private key compose a *key pair*; knowing a public key, it is computationally infeasible to calculate the companion private key. A party can leave its public key available to everyone, e.g. by publishing the key in a public directory; its private key needs to be kept undisclosed. All public key exchange may be done over an insecure channel, i.e. a channel that may be subject to eavesdropping. Public key cryptography therefore requires a Public Key Infrastructure (PKI) to authenticate the parties, generate the key pairs, or distribute, update and revoke the public keys.

Public key cryptography was introduced by Diffie and Hellman [35] in 1976 (and developed further by Merkle [107]), but independently discovered some years earlier by Cocks and Williamson of GCHQ. The Diffie-Hellman key agreement protocol allows two parties to share a secret key over an insecure channel.

One of the greatest problems in a PKI is about how to bind a public key with its legitimate owner – that is, how to be sure that a specific public key belongs to a party and not to an impostor, which would then be able to decrypt messages supposedly sent to that party. If two parties, Alice and Bob (we call them so in the tradition of cryptographic literature), want to exchange their public keys, they could do it over the same insecure channel that is used afterward to swap their encrypted messages. However, if an adversary is able to tamper with communications over the channel, it can make the protection unsuccessful. This is a kind of double identity spoofing, called *man-in-the-middle* attack, in which an adversary stays in the communication channel between two parties and acts with a party as the other party. The parties are deluded that they are talking with each other, while in fact the invisible adversary relays their messages.

The attack is performed as follows. The adversary generates two public/private key pairs $\{P_X, S_X\}, \{P'_X, S'_X\}$. Alice sends her public key P_A to Bob, but the adversary intercepts it, substitutes the legitimate key with its public key P_X, and sends P_X to Bob. Bob sends his public key P_B to Alice,

[1] To avoid confusion, in this thesis we use the acronym MAC for Medium Access Control only in the phrases "MAC layer", "MAC protocol", or "MAC address". In all other contexts, the meaning of MAC must be intended as Message Authentication Code.

but the adversary intercepts and substitutes it with P'_X, which is sent to Alice. As a result, Alice mistakenly believes Bob's public key to be P'_X, and Bob mistakenly believes Alice's public key to be P_X, while both keys are owned by the adversary:

$$\begin{array}{ccccc} \text{Alice} & & \text{adversary} & & \text{Bob} \\ \{P_A, S_A\} & \longleftarrow & \{P'_X, S'_X\}\ \{P_X, S_X\} & \longrightarrow & \{P_B, S_B\} \end{array}$$

From this point on, the adversary intercepts unnoticed any message sent from Alice, decrypts it with S'_X, reads it, re-encrypts it with P_B, and sends the message to Bob which decrypts it with his private key S_B. In the opposite direction, the adversary intercepts any message from Bob, decrypts it with S_X, reads it, re-encrypts it with P_A, and sends the message to Alice which decrypts it with her private key S_A. Therefore, the adversary is able to read any message exchanged between Alice and Bob, while they are unaware of the adversary's presence and think their communications are kept confidential.

One solution to this problem involves a Trusted Third Party, which must be trusted by everyone. The TTP stores the public key of every participant and guarantees on the owner of each key. Depending on the implementation, the TTP is called *Key Distribution Center (KDC)* or *Certification Authority (CA)*. A Certification Authority delivers certificates containing the identity of the key's owner, its public key, the certificate validity dates, and other information; each certificate is signed by the CA, which public key is known a priori by every participant.

For instance, the solution of bestowing a Certification Authority is broadly utilized in the SSL/TLS protocol [148] (on which HTTPS, the secured Internet protocol, is based), IPsec, S/MIME, and others. SSL certificates follow the X.509 standard [50, 63] developed by the International Telecommunication Union - Telecommunication Standardization Sector, and can be delivered by many commercial CAs: RSA Security Inc., VeriSign, ValiCert, and VISA, just to name a few. The public key of each CA is embedded in web browsers and other network applications. Public institutions and government agencies may have their own CAs, too.

However, the existence of a trusted party is a point of fragility of the whole PKI. If the deliver of public keys is done on demand, an adversary could paralyze the whole network by launching a Denial of Service attack against the KDC. Furthermore, by compromising a Certification Authority, the attacker can issue fake certificates for any identity it wishes, to prepare spoofing and man-in-the-middle attacks.

Encryption

To securely send a message, the sender retrieves the receiver's public key, encrypts the message, and sends it to the receiver which can decrypt it with its private key.

Examples of asymmetric ciphers for encryption and decryption are RSA (Rivest-Shamir-Adleman) [135, 136], Knapsack, and ElGamal; other ciphers are instances of elliptic curve cryptography (ECC) applied to canonical algorithms, such as ECC ElGamal. ECC is an approach to the public key problem based on the mathematics of elliptic curves.

Signature

Asymmetric ciphers for signatures are composed of a private and a public part. To sign a message, the sender uses the private algorithm. The receiver of the message then verifies the signature by applying the public algorithm. For simplicity, it is often said that the sender uses its private key to sign while the receiver verifies the signature with the sender's public key.

This is the case of RSA, where the sender generates a hash of the message and encrypts it with its private key. The receiver will use the sender's public key to decrypt the sent hash and check if it matches the recomputed hash. This works because, in a RSA key pair, both the public and private key can be used to encrypt, while the other key is used to decrypt.

Examples of asymmetric schemes to generate digital signatures are Fiat-Shamir, Ong-Schnorr-Shamir, and DSS (Digital Signature Standard) [114] which includes DSA (Digital Signature Algorithm); ECC schemes such as ECNR (Elliptic Curve Nyberg-Reuppel) and ECDSA; and, again, RSA and ElGamal.

2.1.3 Symmetric vs. asymmetric cryptography

Symmetric and asymmetric cryptography has both weak and strong points. Arguments in favor of symmetric cryptography are:

- The data throughput rate is much higher with symmetric ciphers, which also need less computation power.
- For the same level of security, the key size is much smaller with symmetric ciphers. Also, a symmetric digest is smaller than an asymmetric signature.

On the other hand, asymmetric cryptography is superior in some perspectives:

- In symmetric cryptography, the shared key must be kept secret. In asymmetric cryptography, only the private key need to be kept secret, while the public key can (and should) be publicly disclosed.
- Key management is somewhat easier in asymmetric cryptography. To handle a secured message exchange between n parties, the number of

symmetric keys to manage is $O(n^2)$, as there are $\binom{n}{2} = \frac{n(n-1)}{2}$ symmetric keys. Furthermore, if these keys are committed to a Trusted Third Party, this TTP must be *unconditionally trusted* as it is theoretically able to encrypt and decrypt any message from or to any party. Using asymmetric cryptography, the number of keys to manage is just $O(n)$. Only the public keys are entrusted to the TTP, which therefore needs only to be *conditionally trusted*.

- Considering the level of security offered, a public/private key pair may remain unchanged for many sessions. Symmetric keys should be renewed more often (even once per session) to guarantee the same level of security.

In summary, symmetric cryptography is efficient for encryption and data integrity tests, whilst asymmetric cryptography is cogent to generate digital signatures and manage keys. A cleverly designed cryptographic application would exploit the advantages of both schemes: a public key exchange could be used to establish a symmetric key between two parties, while further communications would be encrypted using the symmetric key.

The next chapter provides a classification of the attacks against the routing layer. In Chapter 4 and 5, we show how cryptography can be used to thwart these attacks and enforce security. Chapter 6 offers a dissertation on the available ciphers, considering the requirements and limitations of an ad hoc environment.

Chapter 3

Attacks against ad hoc networks

While a wireless network is more versatile than a wired one, it is also more vulnerable to attacks. This is due to the very nature of radio transmissions, which are made on the air.

On a wired network, an intruder would need to break into a machine of the network or to physically wiretap a cable. On a wireless network, an adversary is able to eavesdrop on all messages within the emission area, by operating in promiscuous mode and using a packet sniffer (and possibly a directional antenna). There is a wide range of tools available to detect, monitor and penetrate an IEEE 802.11 network, such as NetStumbler[1], AiroPeek[2], Kismet[3], AirSnort[4], and Ethereal[5]. Hence, by simply being within radio range, the intruder has access to the network and can easily intercept transmitted data without the sender even knowing (for instance, imagine a laptop computer in a vehicle parked on the street eavesdropping on the communications inside a nearby building). As the intruder is potentially invisible, it can also record, alter, and then retransmit packets as they are emitted by the sender, even pretending that packets come from a legitimate party.

Furthermore, due to the limitations of the medium, communications can easily be perturbed; the intruder can perform this attack by keeping the medium busy sending its own messages, or just by jamming communications with noise.

[1] `http://www.netstumbler.com/downloads`
[2] `http://www.wildpackets.com/products/airopeek`
[3] `http://www.kismetwireless.net`
[4] `http://sourceforge.net/projects/airsnort`
[5] `http://www.ethereal.com`

3.1 Attacks against the routing layer in MANETs

We now focus on attacks against the routing protocol in ad hoc networks. These attacks may have the aim of modifying the routing protocol so that traffic flows through a specific node controlled by the attacker. An attack may also aim at impeding the formation of the network, making legitimate nodes store incorrect routes, and more generally at perturbing the network topology.

Attacks at the routing level can be classified into two main categories: incorrect traffic generation and incorrect traffic relaying [6]. Sometimes these coincide with node misbehaviors that are not due to malice, e.g. node malfunction, battery exhaustion, or radio interference.

3.1.1 Incorrect traffic generation

This category includes attacks which consist in sending false control messages: i.e. control messages sent on behalf of another node (identity spoofing), or control messages which contain incorrect or outdated routing information. The network may exhibit Byzantine [94] behavior, i.e. conflicting information in different parts of the network. The consequences of this attack are degradation in network communications, unreachable nodes, and possible routing loops.

Cache poisoning

As an instance of incorrect traffic generation in a distance vector routing protocol, an attacker node can advertise a zero metric for all destinations, which will cause all the nodes around it to route packets toward the attacker node. Then, by dropping these packets (blackhole attack, see Section 3.1.2), the attacker causes a large part of the communications exchanged in the network to be lost. In a link state protocol, the attacker can falsely declare that it has links with distant nodes. This causes incorrect routes to be stored in the routing table of legitimate nodes, also known as *cache poisoning*.

Message bombing and other DoS attacks

The attacker can also try to perform Denial of Service on the network layer by saturating the medium with a storm of broadcast messages (*message bombing*), reducing nodes' goodput and possibly impeding nodes from communicating. (This is not possible under hybrid routing protocols, where nodes cannot issue broadcast communications [154].) The attacker can even send invalid messages just to keep nodes busy, wasting their CPU cycles and draining their battery power. In this case the attack is not aimed at

[6] Nodes' throughput is composed of two kinds of traffic: control packets and data packets. Here we consider only the former.

modifying the network topology in a certain fashion, but rather at generally perturbing the network functions and communications.

On the transport layer, Kuzmanovic and Knightly [92] demonstrate the effectiveness of a low-rate DoS attack performed by sending short bursts repeated with a slow timescale frequency (*shrew attack*). In the case of severe network congestion, TCP operates on timescales of Retransmission Time Out (RTO). The throughput (composed of legitimate traffic as well as DoS traffic) triggers the TCP congestion control protocol, so the TCP flow enters a timeout and awaits a RTO slot before trying to send another packet. If the attack period is chosen to approximate the RTO of the TCP flow, the flow repeatedly tries to exit timeout state and fails, producing zero throughput. If the attack period is chosen to be slightly greater than the RTO, the throughput is severely reduced. This attack is effective because the sending rate of DoS traffic is too low to be detected by anti-DoS countermeasures.

Another DoS performed on the transport layer is the subtle *jellyfish attack* by Aad et al. [1], that deserves particular attention. Its authors point out that, remarkably, it does not disobey the rules of the routing protocol, even if we may argue that, strictly speaking, this is not always the case. But is indeed true that the jellyfish attack is difficult to distinguish from congestion and packet losses that occur naturally in a network, and therefore is hard and resource-consuming to detect.

This DoS attack can be carried out by employing several mechanisms. One of the mechanisms of the jellyfish attack consists in a node delivering all received packets, but in scrambled order instead of the canonical FIFO order. Duplicate ACKs derive from this malicious behavior, which produces zero goodput although all sent packets are received. This attack cannot be successfully opposed by the actual TCP packet reordering techniques, because such techniques are effective on sporadic and non-systematic reordering.

The second mechanism is the same as that used in the shrew attack, and involves performing a selective blackhole attack by dropping all packets for a very short duration at every RTO. The flow enters timeout at the first packet loss caused by the jellyfish attack, then periodically re-enters the timeout state at every elapsed RTO.

The third mechanism consists in holding a received packet for a random time before processing it, increasing delay variance. This causes TCP traffic to be sent in bursts, therefore increasing the odds of collisions and losses; it increases the RTO value excessively; and it causes an incorrect estimation of the available bandwidth in congestion control protocols based on packet delays.

DoS attacks can also be carried over on the physical layer (e.g. jamming or radio interference); in this case, they can be dealt with by using physical

techniques e.g. spread spectrum modulation [126].

In sum, Denial of Service can be accomplished over different layers and in several ways, and is quite difficult to counteract, even on a wired medium. The topics regarding a full protection against DoS attacks are beyond the scope of this thesis, and therefore are not discussed in detail.

3.1.2 Incorrect traffic relaying

Network communications coming from legitimate, protocol-compliant nodes may be polluted by misbehaving nodes.

Blackhole attack

An attacker can drop received routing messages, instead of relaying them as the protocol requires, in order to reduce the quantity of routing information available to the other nodes. This is called *blackhole attack* by Hu et al. [66], and is a "passive" and a simple way to perform a Denial of Service. The attack can be done selectively (drop routing packets for a specified destination, a packet every n packets, a packet every t seconds, or a randomly selected portion of the packets) or in bulk (drop all packets), and may have the effect of making the destination node unreachable or downgrade communications in the network.[7]

Message tampering

An attacker can also modify the messages originating from other nodes before relaying them, if a mechanism for message integrity (i.e. a digest of the payload) is not utilized.

Replay attack

As topology changes, old control messages, though valid in the past, describe a topology configuration that no longer exists. An attacker can perform a replay attack by recording old valid control messages and re-sending them, to make other nodes update their routing tables with stale routes. This attack is successful even if control messages bear a digest or a digital signature that does not include a timestamp.

[7] Even if a node correctly generates, processes and forwards control traffic, it may act maliciously by not forwarding data traffic. The node thereby breaks the connectivity in the network; however, this connectivity loss is not detected by the routing protocol because control traffic is relayed as required. This type of situation may also be due to wrongly configured nodes: routing capabilities (through IP forwarding) are disabled by default in most operating systems, and need to be enabled manually. Failing to do so effectively causes data traffic not to be routed while control traffic, which is forwarded by action of the routing daemon, is correctly transmitted.

Wormhole attack

The *wormhole attack* [67] is quite severe, and consists in recording traffic from one region of the network and replaying it in a different region. It is carried out by an intruder node X located within transmission range of legitimate nodes A and B, where A and B are not themselves within transmission range of each other. Intruder node X merely tunnels control traffic between A and B (and vice versa), without the modification presumed by the routing protocol – e.g. without stating its address as the source in the packets header – so that X is virtually invisible. This results in an extraneous inexistent $A - B$ link which in fact is controlled by X, as shown in Figure 3.4. Node X can afterwards drop tunneled packets or break this link at will. Two intruder nodes X and X', connected by a wireless or wired private medium, can also collude to create a longer (and more harmful) wormhole, as shown in Figure 3.5.

The severity of the wormhole attack comes from the fact that it is difficult to detect, and is effective even in a network where confidentiality, integrity, authentication, and non-repudiation (via encryption, digesting, and digital signature) are preserved. Furthermore, on a distance vector routing protocol, wormholes are very likely to be chosen as routes because they provide a shorter path – albeit compromised – to the destination. Marshall [103] points out a similar attack, called the *invisible node attack* by Carter and Yasinsac [24], against the Secure Routing Protocol [116].

Rushing attack

An offensive that can be carried out against on-demand routing protocols is the *rushing attack* [68]. Typically, on-demand routing protocols state that nodes must forward only the first received Route Request from each route discovery; all further received Route requests are ignored. This is done in order to reduce cluttering. The attack consists, for the adversary, in quickly forwarding its Route Request messages when a route discovery is initiated. If the Route Requests that first reach the target's neighbors are those of the attacker, then any discovered route includes the attacker.

3.2 Attacks against the OLSR protocol

We now discuss various security risks in OLSR [3, 30]. The aim is not to emphasize flaws in OLSR, as it did not include security measures in its design, like several other routing protocols. While these vulnerabilities are specific to OLSR, they can be seen as instances of what other link state routing protocols, such as OSPF, are subject to.

This section illustrates the principal hazards. More ingenious attacks may be carried over against almost any operating function of the protocol.

It is worth noting that a node can force its election as an MPR by setting the `Willingness` field to the WILL_ALWAYS constant in its HELLOs. According to the protocol, its neighbors will always select it as an MPR. Using this mechanism, a compromised node can easily gain, as an MPR, a privileged position inside the network. It can then exploit its importance to carry out DoS attacks and such like.

Note also that an attacker performing identity spoofing or message replay needs to change the `Message Sequence Number` field of the spoofed or replayed message. Otherwise, nodes that already have received a message with the same originator and MSN (according to their Duplicate Set) will drop the malicious message. Furthermore, accepting the malicious message causes message loss when a legitimate message having the same originator and MSN is received by the victim nodes, and dropped according to the protocol.

3.2.1 Incorrect traffic generation

One way in which a node can misbehave is by generating control messages in a way that is not according to the protocol.

Incorrect HELLO message generation

A misbehaving node X may send HELLO messages with a spoofed originator address set to that of node C (Figure 3.1). Subsequently, nodes A and B may announce reachability to C through their HELLO and TC messages. Furthermore, node X chooses MPRs from among its neighbors, signaling this selection while pretending to have the identity of node C. Therefore, the chosen MPRs will advertise in their TC messages that they provide a last hop to C. Conflicting routes to node C, with possible connectivity loss, may result from this.

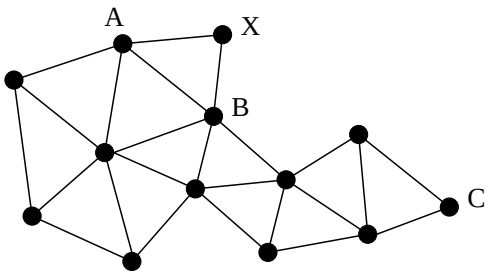

Figure 3.1: Node X sends HELLO messages pretending to be C.

Under identity spoofing, another kind of attack is also possible. A misbehaving node X can set the `Willingness` field to WILL_NEVER on its

HELLO messages sent on behalf of A. According to the protocol, nodes receiving these messages will never choose A as an MPR, which may result in a connectivity loss for some neighbors of A.

We call *link spoofing* the signalization of an incorrect set of neighbors in a control message, and more precisely the signalization of neighbor relationship with non-neighbor nodes. A misbehaving node X may perform link spoofing in its HELLO messages advertising a link with non-neighbor node A, as in Figure 3.2. This will result in C, and the others neighbors of X, storing an incorrect 2-hop neighborhood and therefore selecting a wrong MPR set. In fact, node C will probably select $\{X, D\}$ as its MPR set, instead of the correct MPR set $\{X, B, D\}$, because the first set is smaller. As a consequence, messages originating from E and relayed through the MPR mechanism will not reach node A.

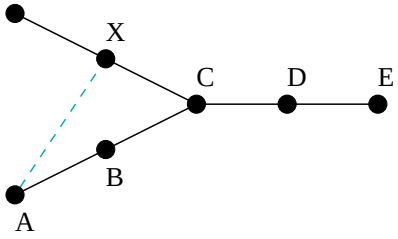

Figure 3.2: Node X sends HELLO messages advertising a fake link with A.

Node X can also misbehave by signaling an incomplete set of neighbors. Depending on their links with other nodes, the ignored neighbors might experience breakdown in connectivity with the rest of the network.

Incorrect TC message generation

TC messages with a spoofed originator address cause incorrect neighbor relationship to be advertised in the network. For instance, node X sends a TC message on behalf of node C, advertising A as a neighbor (Figure 3.3). Node D, upon reception of the TC message, will falsely conclude that C and A are neighbors. For this attack to be successful, the TC message must bear an `ANSN` (Advertised Neighbor Sequence Number) greater than the highest ANSN value referenced to C, as contained in any tuple of D's Topology Set; otherwise D will discard the TC message, according to the protocol.

TC messages with spoofed links have the same effect, and can severely perturb the network topology as stored by legitimate nodes.

Node X can also simply generate HELLOs, perhaps be selected as an MPR by its neighbors, but refuse to generate TC messages or generate TCs signaling an incomplete set of nodes. The OLSR specifications require that X includes at least its MPR selectors in its TCs; if this requirement is not

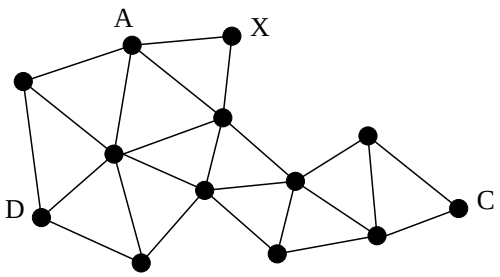

Figure 3.3: Node X sends TC messages pretending to be C.

fulfilled, some nodes may not have their link state information disseminated throughout the network and be disconnected.

Node X, behaving incorrectly, can also send TC messages without being an MPR. The protocol specifications state that only MPRs generate TCs; however, there is no way of detecting whether the originator of a TC message is an MPR of some node or not.

Incorrect MID/HNA message generation

A misbehaving node X can generate wrong MID/HNA messages, declaring interfaces that are not their own (link spoofing), or falsifying the originator address of the message (identity spoofing) so that it apparently declares interfaces that are not their own. In this case, nodes will have problems reaching these interfaces.

ANSN attack

The misbehaving node may listen to a TC message from node A and record the ANSN of the message; then it sends a TC with a spoofed originator address of node A, and an ANSN much greater than the value recorded. According to the protocol specifications, nodes will ignore further TC messages from A, because these messages bear a smaller ANSN as that recorded in the Topology Set, and therefore such messages are considered as arrived out of order. We call this an *ANSN attack*. If no further action is taken by the attacker, the ANSN attack is effective until the ANSN of node A reaches the value of the ANSN in the spoofed TC.

This attack can be spotted as the spoofed TC bears an ANSN which is much higher than that of the latest genuine TC message received from A (the higher the difference between the two ANSNs, the longer TCs from A are ignored). However, the misbehaving node may perform this attack repeatedly, by forging each time spoofed TC messages with a slightly greater ANSN.

3.2.2 Incorrect traffic relaying

If control messages are not properly relayed, network malfunctions are possible.

Blackhole attack

If a node fails to relay TC messages, the network may experience connectivity problems. In networks where no redundancy exists (e.g. in a strip), connectivity loss will surely result, while other topologies may provide redundant connectivity.

If MID and HNA messages are not properly resent, additional information regarding multiple nodes interfaces and connections with external networks may be lost.

Replay attack

As previously said, replaying old control messages in the network causes nodes to record stale topology information. A control message cannot be replayed "as is" or it will not be accepted by nodes that already received it, because of the MSN. Therefore the attacker needs to increase the MSN of the message, causing possible message loss. For a TC, the attacker must increase the ANSN too, indirectly causing an ANSN attack. Replayed HELLOs may have a lesser impact, because link state advertised in HELLOs must be given in a well-defined order (see Section 9.1).

Wormhole attack

An extraneous $A - B$ link can be artificially created by an intruder node X by wormholing control messages between A and B (Figure 3.4). A longer wormhole can also be created by two colluding intruders X and X' (Figure 3.5).

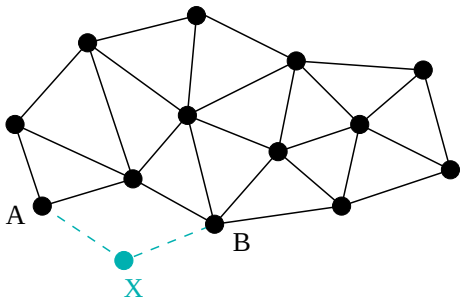

Figure 3.4: A wormhole created by node X.

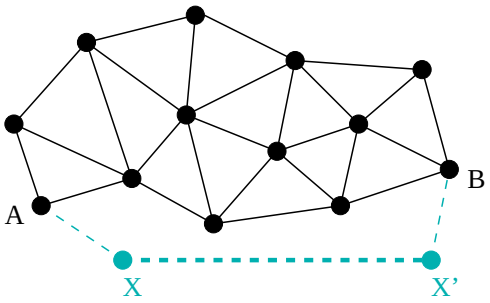

Figure 3.5: A longer wormhole created by two colluding nodes X and X'.

To successfully exploit the wormhole, the attacker must wait until A and B have exchanged sufficient HELLO messages (through the wormhole) to establish a symmetric link. Until that moment, other tunneled control messages would be rejected, because the OLSR protocol specifies that TC/MID/HNA messages should not be processed if the relayer node (the last hop) is not a symmetric neighbor. However, once created, the $A - B$ link is at the mercy of the attacker.

MPR attack

The "first transmit rule", described in the OLSR specifications, states that a node receiving a message in MPR flooding checks if the sender is its MPR selector. If so, the node retransmits the message. If the sender is not an MPR selector of the node, the latter will never retransmit the message. While this rule is established for performance reasons (to avoid messages traveling on large loops in dense networks) it could be exploited to impede the correct relaying of control messages.

We call the related misbehavior an *MPR attack*. Consider the following scenario (Figure 3.6): node A sends a message to its neighbors B and X, where B is an MPR of A, X is not an MPR, and C is an MPR of B. The misbehaving node X does not select its MPR set properly, and retransmits the message (even if it is not supposed to) which is received by C. Node B retransmits the message to C. The crucial point is that C, even being an MPR, will not relay the message because C has already received it from X.

3.3 Summary of routing attacks

All the depicted attacks are possible at a theoretical level; most of them are very easy to implement and require even less energy and effort than running a protocol-compliant node. Table 3.1 summarizes the effect of each attack on each particular function of an OLSR network.

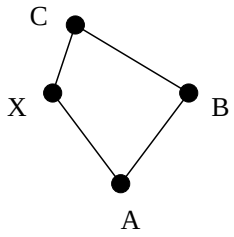

Figure 3.6: Node X performs an MPR attack.

			Conflicting routes	Connectivity loss	Message loss
Incorrect traffic generation	Incorrect HELLO generation	ID spoofing	×	×	×
		link spoofing	×	×	
	Incorrect TC generation	ID spoofing	×	×	×
		link spoofing	×	×	
	Incorrect MID/HNA generation		×	×	×
	ANSN attack			×	×
Incorrect traffic relaying	Message tampering		×	×	
	Blackhole attack			×	×
	Replay attack		×	×	×
	Wormhole attack			×	
	MPR attack			×	×
Message bombing and other DoS				×	×

Table 3.1: OLSR attacks and their effects on the network.

Concerning the realism of these attacks (real attacks that have been observed against existing networks), there is no or very little data available. This is probably due to the fact that ad hoc networks are in practice still used in limited environments such as warfare operations, search and rescue missions, and research centers; while the mainstream architecture for a wireless network is BSS, with "hot spots" offered by various ISPs in airports, train stations, museums, restaurants, and other public places.

It is indeed true that some offensive behavior (e.g. DoS) can also successfully be carried out at the physical or transport layer. However, in our opinion, it is necessary to foresee these routing attacks, otherwise when these attacks are carried out (and certainly they will be) we will be unable to recognize them as such.

Chapter 4

Security in ad hoc networks: basic mechanisms

Wireless transmissions utilize a shared medium – the air – that is virtually accessible to anybody at any time. As it is not possible to limit access to the medium, the only way to protect messages is to use cryptography to sign or encrypt the exchanged data.

The IEEE 802.11b standard includes a scheme called WEP (Wired Equivalent Privacy) to secure communications. It uses the RC4 stream cipher coupled to an Initialization Vector for encryption, and the CRC-32 checksum for integrity check. WEP employs a 40-bit or 64-bit secret shared key, providing no infrastructure for key management. As its name implies, WEP offers a protection similar to that of an unsecured wired network, and therefore a quite low level of security (the AirSnort program can easily crack WEP keys). Despite its weakness, WEP may be considered useful as a deterrent against casual snoopers.

The vulnerabilities of WEP have been fixed in WPA (Wi-Fi Protected Access). WPA uses IEEE 802.1X authentication, providing port-based network access control capability, with a standard EAP (Extensible Authentication Protocol) [15].

A stronger security system is specified in the IEEE 802.11i standard [74], also known as WPA2. The WPA2 standard adds the AES (Advanced Encryption Standard) security protocol to IEEE 802.11.

4.1 Protection of the routing protocol

In general, the desired security for the routing mechanism concerns integrity (less often non-repudiation) and service availability. Therefore, when talking about protecting routing control messages, we mostly consider how to generate and verify digests or digital signatures. Encryption is often left aside, because is more time- and power-consuming, and because confidentiality is

not usually required, as routing information is not secret. (However, this is not always true. In the case of military applications, routing information may be tactical information of primary importance; for instance, it could help enemies identify and locate their targets on a battlefield.)

4.2 State of the art

The protection of the network can be obtained through cryptographic tools such as IPsec, or via dedicated solutions. In the literature there are many proposals for secured routing protocols that provide message integrity (through digests) and/or sender authentication (through signatures). Several of these are modifications of standard non-secure routing protocols. Other protocols provide security in a different way.

4.2.1 IPsec

IPsec [87, 98] is an IETF standard that incorporates various security services for the IP layer. The IPsec framework provides authentication and encryption of data packets [85, 86], maintenance of security associations (with shared secret keys) between peers [105], and manual or automated key management [61].

It has been pointed out [137, 128] that, in general, it is impossible (or at least impractical) to use IPsec to secure routing protocols in MANETs, because IPsec assumes that a security association between pairs of nodes already exists; and, of course, this is not the case in an emerging ad hoc network. The fundamental problems of IPsec with respect to securing OLSR are detailed in the following.

First, the automated key exchange, which also provides the automated timestamp exchange for protection against replay attacks, assumes that the parties can reach each other. This is not the general case with the secured version of OLSR, because messages must be authenticated before being accepted; hence a node which arrives in the network accepts no packets, and has no routes. Two remedies are possible: either using pure flooding for the messages, or changing the OLSR specifications as we show in Section 6.3.7.

Second, IPsec protects the packet itself, while the granularity of the protection that we propose is the message. For technical reasons, it is not possible to sign or generate a digest of a whole OLSR packet, nor it is desirable to do so; see Chapter 5. One remedy could be to forbid any change of the packets in transit, so that each message would go in a different packet. This would have a certain cost on wireless networks, where overhead per-packet on the MAC layer is large in some cases. Furthermore, this might not be sufficient and other requirements, such as the use of tunnel mode, should be made, along with technical necessities such as using the TTL field of the IP packet instead of the OLSR packet.

Third, the current IPsec implementations support essentially symmetric keys. However this may change, as a recent IETF draft proposes asymmetric signatures [157].

Last, managing a group key or a set of group keys in the context of an ad hoc network is a different problem than just the issues in the multicast or group key management protocols such as GKMP (Group Key Management Protocol) [62] or MIKEY (Multimedia Internet KEYing) [5]. This because in an ad hoc network all nodes are senders and all nodes are receivers of the OLSR protocol messages. Network splits or merges also need to be managed.

In general, few IPsec security schemes may be used, and even these need significant modifications such as pure flooding for message transmission. This comes at the cost of security granularity and performance: for instance, in a network of n nodes, if each node creates and transmits a session key with every other node using pure flooding, the cost is $O(n^3)$. This would also be likely to result in using IPsec on the limits of their domain of applicability. It is also worth noting that the complexity of many IPsec protocols is already greater than the complexity of the OLSR protocol itself.

4.2.2 Routing protocols using digests or signatures

SRP (Secure Routing Protocol) [116], by Papadimitratos and Haas, is built on the basis of DSR, and requires a security association between each pair of communicating nodes. When initiating a route discovery, a node inserts in the SRP header of the query packet the following information: a sequence number, a nonce, and a MAC. The MAC is a keyed hash computed on the IP header, the sequence number, the nonce, and the shared secret key. In the route reply message, the MAC includes also the route.

Unfortunately, it has been observed that a security flaw makes SRP defective [103]. In a route discovery, a malicious intermediate node may not append its address to the route request and reply messages (as it is supposed to do). As a consequence, the originator of the route discovery validates a route which in fact does not exist.

SLSP (Secure Link State routing Protocol) [117], by the same authors of SRP, is a proactive secure routing protocol that makes use of asymmetric cryptography to protect control messages. The security mechanisms of SLSP are committed to the Neighbor Lookup Protocol, which maintains a mapping of IP and MAC (hardware) addresses extracted from overheard frames; the protocol uses this mapping to identify discrepancies such as multiple addresses.

SAODV (Secure Ad hoc On-demand Distance Vector routing) [164] is the secured version of AODV. RREQ and RREP messages are signed by a sending node, and the signature is verified by intermediate nodes before

forwarding the message. Optionally, the RREQ message bears a second signature which, if an intermediate node wants to reply with a RREP, is utilized in the reverse route.

ARAN (Authenticated Routing for Ad hoc Networks) [137] is an on-demand routing protocol which requires a TTP certificate server. Nodes request a certificate from the certificate server, then they sign all generated messages.

A polyvalent technique based on certificates [128] consists in appending a module, called MAE (MANET Authentication Extension) to each routing message. This technique uses threshold cryptography to provide a distributed and self-organized certification service. The MAE protocol can be applied to DSR, AODV, OLSR, TBRPF, and possibly other routing protocols.

Ariadne [66] is based on DSR for the routing architecture and on TESLA (Timed Efficient Stream Loss-tolerant Authentication) [125, 124] for the authentication mechanism. TESLA guarantees the integrity of communications by adding a MAC in each message, and provides some form of authentication by *one-way key chains* (also called *one-way hash chains*), computed by agreeing on a hash function h. A node, during initialization, chooses a random number x and computes the list of values $h_0, h_1, h_2, \ldots, h_n$, where $h_0 = x$, and $h_i = h(h_{i-1})$ for $i \leq n$. The node will afterwards publish these keys in reverse order from h_n to h_0, following a predetermined schedule. Before sending a message, the sender node estimates an upper bound T_u on the end-to-end network delay, and computes the MAC on the message with a key h_i which will not be disclosed until after the delay. Upon reception of the message, the receiver node verifies that the key h_i is still secret, then waits until h_i is disclosed by the sender. After that, the receiver node authenticates h_i. The receiver node is also able to authenticate a value h_{i-1} contained in a further message by verifying that $h(h_{i-1}) = h_i$, or authenticate any h_{i-j} by applying j times the hash function i.e. $h^j(h_{i-j}) = h_i$.

Ariadne uses symmetric encryption for efficiency reasons, but its authors also provide a modification to include a Key Distribution Center for key authentication. In Ariadne, a node originating a route discovery broadcasts a Route Request message, containing a time interval greater than T_u and protected with a MAC. Each intermediate node that receives the Route Request verifies if the associated key is still undisclosed according to the time interval; if so, the node appends its MAC to the message and forwards it. The target node performs the same tests, then sends a Route Reply to the originator node via the reverse path. Every intermediate node that receives the Route Reply waits until it can disclose its key according to the time interval, then appends its key to the message and forwards it. Finally, upon

reception of the Route Reply, the originator node checks that all included keys and MACs are valid.

TIK (TESLA with Instant Key disclosure) [67] is a protocol designed for defense against wormhole attacks. TIK uses what its authors call a *packet leash*, i.e. a piece of information added to a packet to restrict the packet's maximum allowed transmission distance (geographical leash) or lifetime (temporal leash). All nodes must have tightly synchronized clocks. Key authentication is accomplished using hash trees, which are an optimization of the one-way hash chains discussed above.

A sender node S generates the MAC, denoted by $H(M, k_i)$, of a message M using key k_i. Key k_i has disclosure time t_i (in the future) and can be authenticated by the hash tree value h_i. The MAC is included in the header part of the packet. Before sending the packet, S estimates an upper bound on the arrival time of the packet to the receiver node R, and appends the key k_i to the packet: $S \rightarrow R : \{H(M, k_i), M, h_i, k_i\}$. Upon arrival of the MAC, R verifies that S has not yet started to send k_i, based on the disclosure time t_i. If this is true, R authenticates the key k_i using h_i, and verifies $H(M, k_i)$.

SEAD (Secure Efficient Ad hoc Distance vector routing) [65] is based on the design of DSDV. Nodes authenticate with each other by using hash chains. A node chooses a random number x and computes $h_0, h_1, h_2, \ldots, h_n$, where $h_0 = x$, and $h_i = h(h_{i-1})$ for $i \leq n$ as said before. The value h_n is firstly distributed either by a Certification Authority, or by direct exchange (using symmetric cryptography) between nodes, or by any other infrastructure for key distribution. Afterwards, the node includes these values in its messages, one for each message, and in reverse order from h_{n-1} to h_0. Receiving nodes , knowing h_i, authenticate the value h_{i-1} contained in a further message by verifying that $h(h_{i-1}) = h_i$. With this protocol it is still possible for an attacker to tamper with messages while they are in transit.

4.2.3 Other solutions

Hu et al. propose RAP (Rushing Attack Prevention) [68], a generic component for secure route discovery in reactive routing protocols. RAP is aimed at protecting the network against rushing attacks. RAP contains three mechanisms: secure neighbor detection, secure route delegation, and randomized Route Request forwarding.

Secure neighbor detection is accomplished by observing the challenge-response delay, to evaluate the distance to a node and verify if the node can be a neighbor. Secure route delegation is done through exchange of Route Delegation / Accept Delegation messages between verified neighbors, before any forwarding of a Route Request. Furthermore, instead of forwarding the first received Route Request, a node collects a number of Route Requests

and then randomly chooses the one to be forwarded.

RAP uses HORS [133] and the constructions of BiBa [123] as a fast one-time signature mechanism. HORS, designed by Reyzin and Reyzin as an improvement on BiBa, is a one-time signature scheme i.e. a signature scheme that can be used once or a small number of times. The characteristics of HORS are a short signature and very fast signature and verification, hence its suitability for multicast and broadcast authentication.

SAR (Security-Aware ad hoc Routing) [163] is a modification of a traditional, non-secured route discovery protocol (like AODV, DSR, or ZRP) to include the security level of a node into routing metrics. The nodes are organized in a trust hierarchy; a number is associated with each privilege level and represents the security/importance/capability of a node. RREQ and RREP packets are encrypted, and a cryptographic key is assigned to each level; this can be obtained by setting the key length so that it is proportional to the requested level of security. In this manner, packets are routed only through safe nodes; nodes without the required security rank cannot even read the control packets and must therefore drop them.

Lee et al. [96] suggest securing DSR by adding two new control messages: Route Confirmation Request (CREQ) and Route Confirmation Reply (CREP). These messages are used as a confirmation in the RREQ/RREP route discovery mechanism.

When an intermediate node replies with a RREP, the protocol requires that it sends a CREQ to its next-hop node towards the destination. Then the next-hop node, if it has a route to the destination in its cache, replies with a CREP to the source of the RREQ. Hence, the source node can verify the validity of the obtained route by comparing the RREP with the CREP.

This mechanism does not use cryptography, and consequently is still vulnerable to message tampering and identity spoofing.

Buttyán and Hubaux propose two mechanisms to improve service availability on an open network. Packet forwarding consumes the battery energy of a node; therefore, a node may be tampered with, or simply switched off, by its participating user so as not to provide this service.

The first mechanism [21] introduces an abstract currency called *beans*. This currency is paid by the originator of a packet to the forwarding nodes for the forwarding service (Packet Purse Model), or exchanged for a packet which will be sold to the next hop for a higher price (Packet Trade Model). The motivation of nodes to earn stimulates cooperation and avoids node selfishness. A PKI is used to guarantee authentication and establish secure communications.

In the second mechanism [22] a tamper resistant security module, embedded in each node, maintains a *nuglet counter* which is decreased when

the node originates a packet and increased when the node forwards a packet. A node's cooperation is ensured by requiring that the value of the counter must remain positive.

Privacy is often not of primary importance in routing, and secured routing protocols are more focused on providing other security goals; for this reason it is worth mentioning the PPR (Privacy Preserving Routing) protocol [154], which is aimed at protecting nodes' identities.

4.3 Secured versions of OLSR

Among the security solutions examined up to now, only a few could be applied or adapted to OLSR. For instance, SAODV is based on AODV and is aimed at protecting the route discovery mechanism, which in a proactive routing protocol such as OLSR would not make sense. The purpose of TIK is primarily to provide defense against the wormhole attack; furthermore, this protocol requires a tight synchronization between nodes, which is not easy to obtain in an ad hoc environment. The MAE architecture can be applied to OLSR, as well as to other routing protocols; however, our aim is to find a dedicated security architecture that can be interfaced with the functioning of OLSR so that the OLSR mechanisms are fully exploited. For instance, a clever use of the OLSR Duplicate Set can permit a loose synchronization: we illustrate this in our security solution for OLSR, discussed in Chapter 5.

In this section we give an overview of other security solutions explicitly designed for OLSR, as found in the literature.

4.3.1 Packet protection

Secure OLSR [60], a proposed technique for securing OLSR, involves protection and hop-by-hop check of a whole packet. A digest is computed by the forwarder node, added to the OLSR packet, and verified by the next-hop node. This allows the digest to encompass mutable fields such as the `Time To Live` or the `Hop Count`. The digest is added in the form of a control message, and includes a timestamp. The algorithm used to digest the packet is SHA-1 with a secret shared key. Time synchronization is done through a challenge-response mechanism, via dedicated control messages.

SOLSR [64] protects the traffic by adding a packet signature, while using hash chains to secure the `Time To Live` and `Hop Count` mutable fields. It also implements a defense against the wormhole attack. A node sends probe packets to measure their travel time, from which it can compute the travel distance. Then the node evaluates this distance: if it is greater than the transmission range, the message may have been tunneled through a wormhole.

4.3.2 Message protection

There is a recent proposal of a secured OLSR protocol [78] that uses both symmetric and asymmetric keys. Nodes mutually authenticate using public key cryptography, performing re-authentication when moving to another neighborhood. During the authentication, nodes share two symmetric keys: a *circle key*, utilized among neighbors only, and an *ad hoc key*, utilized in the whole network. A MAC is computed with the circle key and added to control messages to ensure message integrity. Nodes periodically renew both keys, and the new key is distributed after being encrypted with the old key; for this purpose, the new ad hoc key is included in TC messages.

4.3.3 Trust Metric Routing

Winjum et al. [159, 160] propose an extension for OLSR that uses Trust Metric Routing. The concept of Trust Metric Routing is to divide the network into different security domains, where only nodes belonging to the same domain share intra-domain security parameters like keys and such. A user may choose to route packets through trustworthy routes, which are fully contained in the same domain, or ordinary routes, which spread over multiple domains. To this end, two routing tables are maintained by each node: an ordinary routing table calculated with the standard shortest path algorithm, and a trustworthy routing table calculated using trust parameters and showing only intra-domain routes. The information about the trust level of a link or route is integrated and exchanged in control messages.

Chapter 5

The OLSR signature message

We design here an infrastructure [4, 2] to protect OLSR. A prototype of this infrastructure has been built for an INRIA contract with the DGA CELAR, the French government agency for weaponry. This framework can operate with either symmetric or asymmetric keys. To prevent malicious nodes from injecting incorrect information into the OLSR network, an additional security element is generated by the originator of each control message and transmitted with the control message. For the sake of simplicity, in this chapter we call the additional element a "signature" even if in the case of a shared symmetric key it should, more properly, be called a "digest". A timestamp is associated with each signature in order to estimate message freshness. Thus, upon receiving the control message, a node can determine if the message originates from a trusted node, or if message integrity is preserved.

Signatures are, inherently, separate entities from OLSR control traffic: while OLSR control messages answer the purpose of acquiring and distributing topological information, signatures serve to validate information origin or integrity. For this reason we implement the signature as a separate type of OLSR message (called SIGNATURE message), instead of appending the timestamp and the signature to the control message. The resulting signature message is considered and handled like any other OLSR standard message. Furthermore, while this implementation slightly increases the total message size, it does not involve considerable modifications to the standard OLSR protocol as it uses the standard format for the control messages.

5.1 Specifications

For each control message (HELLO, TC, MID, or HNA) generated, a corresponding SIGNATURE message is generated, and sent in the same packet containing the control message, immediately before it. Signatures are used by a receiving node to authenticate the corresponding OLSR control message: every control message without a matching corresponding signature is

dropped.

In our architecture, a signature purports to a message, not to a whole packet. It is not possible to sign or digest a whole OLSR packet because it may change in transit between one node to another. This is because a packet may contain TCs, which are flooded in the network, as well as HELLOs, which are not forwarded further. Hence after a few hops the packet might no longer bear a valid signature, because it was computed on the original packet.

A remedy to the payload change problem would otherwise be to check the signature on a hop-by-hop basis (with re-computation of the signature at each hop) instead of an end-to-end check. However, signing a OLSR packet would also have profound implications with respect to accountability in the case of a compromised node: as many nodes repeat the TC messages that are diffused by MPR flooding, if a message is found to be incorrect, any of the nodes which repeated it might be a compromised node. When the authentication is per packet, it may only be deduced that the compromised node is part of the (previously) trusted network. When the authentication is per message, the node originator of the message is easily identified as the origin of the faulty information. For these reasons, the logical conclusion for the choice of a packet signature algorithm would be a digest computed with a symmetric shared secret key for the whole group of nodes, protecting the messages on a per-group basis, and not offering the possibility to check which node sent a specific message. This very architecture has in fact been proposed [60]. The advantage of this option is that it makes it possible to include the TTL and Hop Count, which are mutable fields, in the digest.

We decided on the choice of signing single messages also because it permits switching to an asymmetric algorithm with minimal changes and effort. Moreover, this architecture is better compatible with the standard OLSR, as signature checking can be turned off if bandwidth is needed and security requirements become looser.

The control message and its SIGNATURE message are sent in the same OLSR packet in order to simplify handling of the messages: the packet contains first the SIGNATURE message, then immediately after the control message it purports to. (If these messages were not sent in the same packet, their order of arrival could not be guaranteed. Therefore each node would need a buffer to temporarily store them after reception, before trying to couple them.)

The difficulty posed by handling long packets that exceed the MTU is solved as follows. The control message may be fragmented if necessary, so that the control message and its SIGNATURE are smaller or equal to the MTU of the network. If the control message is fragmented, an independent SIGNATURE message must be computed and assigned to each fragment. Fragmentation may also be used for messages that are waiting in the relaying queue, in order to insert these messages in the packet ready to be sent. Note that is not strictly necessary to consider the MTU of the

network (i.e. the minimum bound of the MTUs of all the link pairs in the network): in fact, for fragmentation of HELLO messages, only the MTU size of the sender-receiver link needs to be considered, because HELLOs are not forwarded further. For simplicity, however, we always consider the network MTU in the fragmentation rule.[1]

A unique OLSR packet may contain more than one pair of control message and SIGNATURE message, provided that the payload contains a SIGNATURE immediately before its companion control message, in this exact order.

5.1.1 Format of the signature message

The SIGNATURE message is encapsulated and transmitted as the data portion of the standard OLSR packet format described in Section 1.4.4.

The `Message Type` field is set to the SIGNATURE constant value; this value may also include information about the cryptographic primitives and keys to use. The `Time To Live` and `Vtime` fields are set to the values of the `Time To Live` and `Vtime` fields of the message with which the signature is associated. The other fields of the message header are set as usual.

Extended version

An old version [2] of the SIGNATURE message is shown in Figure 5.1. The message carries a `MSN Referrer` field in order to identify bijection between a control message and its SIGNATURE message.

The `Sign. Method` field specifies which method, among a predefined set, is being used to generate the signature. This includes information about keys, cryptographic functions, and timestamp algorithms.

The `Reserved` field is used for padding, to make all fields 32 bit aligned. It is set to 0 and reserved for future use.

The `MSN Referrer` field of the SIGNATURE message contains the value of the `Message Sequence Number` of the control message with which this signature is associated. The correspondence achieved by the Message Sequence Number is unique only if possible overflow and wraparound of the 16-bit field is disregarded; however this is not a problem, since a node uses further signature (and timestamp) verification to check the correspondence between the control message and the signature message.

The `Timestamp` and `Signature` fields are the same as in the actual version of the message.

The approach implemented in the previous version makes it unnecessary to send the SIGNATURE message and its associated control message in the same packet, as the messages could be reordered and re-associated

[1] In IEEE 802.11b a data link frame may carry up to 2304 bytes. This gives a MTU of 2272 bytes for IPv4 addresses, not considering IP, UDP, and OLSR packet headers [127].

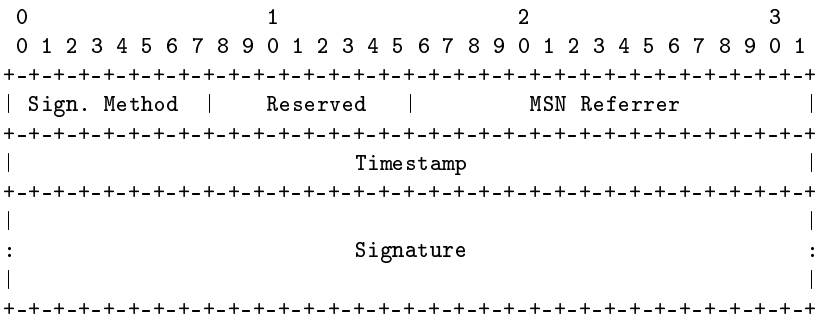

Figure 5.1: Old version of SIGNATURE message format.

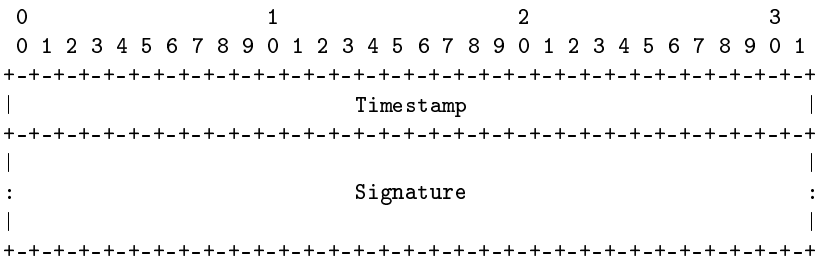

Figure 5.2: SIGNATURE message format.

later. However, this means that every node would need to store the received messages (control and signature messages) in a buffer. This requires more system resources and is more prone to failure and DoS attacks (regarding control messages whose signature is lost, or vice versa). Furthermore, delay is unfavorable when a message and its signature are not aggregated in the same packet [158]. This approach was hence abandoned in favor of the actual simplified version.

Simplified version

The actual format of a SIGNATURE message is specified in Figure 5.2.

The `Timestamp` field contains the timestamp itself, measured in seconds. This is the timestamp of both the SIGNATURE message and the associated control message. For compatibility reasons, the timestamp is 32 bits long and represents the standard Unix time, which is encoded in a 32-bit signed integer[2] data type. The Unix time measures the time elapsed in seconds since 00:00:00 UTC on January 1, 1970.

The current time is obtained from the node's internal BIOS clock. The BIOS clock has a linear drift of about 1 sec/day, which can therefore be

[2]Or a 64-bit signed integer in the newer versions of Unix.

corrected via an algorithm. See Section 7.2 for an empirical study on time synchronization techniques.

The `Signature` field contains the signature, computed on the sequence of bits made from the following fields:

- the message header (80 bits) of the control message, excluding the `Time To Live` and `Hop Count` fields. These fields are not considered in the signature computation because they are modified while the message is in transit (the `Time To Live` is decreased by 1 and the `Hop Count` is increased by 1 at each hop), and subsequently the signature of the message would be invalidated;
- the control message (which has a variable size);
- the message header (80 bits) of the SIGNATURE message, excluding the `Time To Live` and `Hop Count` fields;
- the `Timestamp` field (32 bits).

5.1.2 The timestamp

The criterion to verify whether a timestamp is stale is $|\texttt{Timestamp} - t_0| \leq \Delta t$, where t_0 is the current time at the receiving node and Δt is the accepted value for discrepancy, including the difference in the synchronization of clocks.

A strict clock synchronization of the nodes is not necessary; the timestamp is used to disambiguate possible wraparound of the Message Sequence Number. The synergy of timestamp and Message Sequence Number in every message is used to check the freshness of the message, and wraparounds of Message Sequence Number are a rare event. In fact, counting a time interval of 2 seconds for HELLOs and 5 seconds for the other control messages (standard OLSR values), and the `Message Sequence Number` field being 16 bits long, wraparounds of the MSN occur no more frequently than every 16 hours for the standard OLSR or 8 hours for the secured OLSR.

However, the synchronization must not be coarser than the lifetime of the Duplicate Set; in fact, a Duplicate Tuple is deleted from the Duplicate Set when it is 30 seconds old (DUP_HOLD_TIME constant), and a node may be subject to the possible replay of a message that has the same MSN as that of a deleted Duplicate Tuple.

In our CELAR implementation, we let $\Delta t = \text{DUP\_HOLD\_TIME}/2$.

5.1.3 The signature algorithms

Our security architecture [4] relies on the use of asymmetric cryptography. An offline Certification Authority has the duty of assigning an identity-based key pair for each participating node. Before joining the network, a node contacts the Certification Authority through a secure channel, and obtains

Key type	Elliptic curve	Signature size	Security
Global key (CC)	$n_g = 2^{174} + 2^{115} + 1$ $p_g = (2^{339} n_g)^2 + 1$ $y^2 = x^3 - x$	$(U, V) \in E(F_{p_g})^2$ (4104 bits)	$\sim$ 80 bits $\sim$ RSA 1024
Local key (BLS)	$n_l = 2^{14} + 2^5 + 1$ $p_l = (2n_l)^2 + 1$ $y^2 = x^3 - 4x$	$s \in E(F_{p_l})$ (64 bits)	very low

Table 5.1: Elliptic curve parameters for global and local keys.

Key type and operation	486	P3	P4
Global key, signature (CC)	$18.30 \cdot 10^3$	$0.51 \cdot 10^3$	$0.25 \cdot 10^3$
Global key, verification (CC)	$77.30 \cdot 10^3$	$2.12 \cdot 10^3$	$1.08 \cdot 10^3$
Local key, signature (BLS)	30.00	1.10	0.48
Local key, verification (BLS)	43.00	1.57	0.72
Local key, Weil pairing	7.83	0.28	0.12

Table 5.2: Benchmarks for operations on global and local keys (msec).

a *global key*. The node also generates a key pair, and diffuses its public key (*local key*) to the network via a specific key exchange protocol: it originates Key Distribution messages, signed with its global key, that are spread by pure flooding. From this point on, the node uses its local key to sign its control messages.

This implementation utilizes identity-based Cha-Cheon signatures [25] (pairing based) for the global keys, and Boneh-Lynn-Shacham short signatures [17] for the local keys. In both cases, a Weil pairing is used on supersingular elliptic curves of embedding degree $k = 1$, with the family of curves proposed by Koblitz and Menezes [89]. The parameters are shown in Table 5.1. The implementation has been tested on an Intel i486 133 MHz, on an Intel Pentium III 1 GHz, and on an Intel Pentium 4 2.8 GHz, giving the results shown in Table 5.2. These solutions must be seen as prototypes, as the figures show that size of global keys is sufficient to ensure some degree of security but the computation is slow, while local keys have fast computation times but small size (insecure).

5.1.4 Applicability to control messages

It may be discussed whether it is appropriate to sign every type of control message, or just some types. In the first case, there would obviously be a larger overhead. As the primary purpose is to protect the network topology, it is mandatory to choose to associate a signature to control traffic messages

(HELLO and TC) only. We decided nonetheless to sign even the other OLSR control messages (MID and HNA), in order to avoid false information about multiple interfaces being spread over the network.

5.1.5 Optional features

As previously said, the `Time To Live` and `Hop Count` fields in the message headers cannot be included in the signature computation, since these fields change at each hop of the message and this would interfere with the correct verification of the signature by the receiving node. This unfortunately leaves the door open to an attack where an adversary relays tampered messages whose TTL has been set to 0 or 1 or, more generally, to a lower value than the original. This weakness can be overcome by ignoring the `Time To Live` field, and referring to the `Timestamp` field (which is protected by the signature) to limit the forwarding radius of the message.

We recall that, in order to make flooding more robust, it is possible to allow each node to select two (or even more) MPRs to cover all its 2-hop neighborhood, by setting an appropriate MPR_COVERAGE constant. Redundant MPR coverage will be, of course, at the expense of MPR flooding efficiency. This remedy can also be used to cure blackhole attacks and incorrect MPR selection from malicious nodes; incorrect selection of MPRs is also sometimes performed by legitimate nodes as an effect of wrong topology spread by malicious nodes, as explained in Section 3.2.

5.1.6 Interoperability with standard OLSR

This security architecture is not interoperable with the standard OLSR. Non-secured nodes, i.e. the nodes which do not have the ability to check the signature (because of limited computing power or non-knowledge of the key), may simply drop SIGNATURE messages upon reception. However, their unsigned control messages would be dropped by secured nodes. This means that secured nodes could not reply to HELLO messages from non-secured nodes, therefore no symmetrical link and subsequently no MPR relationship could be created between secured and non-secured nodes. As a result, there would be two disjoint networks, one composed of secured nodes and the other composed of non-secured nodes. The secured nodes would totally ignore messages from non-secured nodes, while non-secured nodes would process messages from secured nodes but only to create asymmetrical links which disappear shortly thereafter. The coexistence of the two networks would only have the effect of producing a larger bandwidth consumption.

5.2 Modifications to the standard OLSR protocol

Securing the OLSR protocol involves modifying some parts of its basic functioning.

5.2.1 Sending a signed control message

In brief, to compute a signature corresponding to a control message, the following protocol is used:

1. the node creates the control message;
2. the node retrieves the current time, and writes it in the `Timestamp` field;
3. the node computes the signature, and writes it in the `Signature` field;
4. the node puts the SIGNATURE message and the control message in the packet, in this exact order.

Then, the node sends the packet, or repeats the protocol for another control message before sending the packet.

5.2.2 Changes to the Duplicate Set

The Duplicate Set of the standard OLSR is modified to include a new field *D_timestamp*. This field stores the value of the `Timestamp` field, once the matching between the SIGNATURE message and the control message has been found. The *D_timestamp* field is filled with the same value for the control message and its SIGNATURE. Incoming messages are recorded in the Duplicate Set as usual.

5.2.3 Receiving and checking a signed control message

Upon receiving a control message with its SIGNATURE message, a node processes both. The protocol is outlined as follows:

1. the node processes the SIGNATURE message, checking the timestamp, and keeps the SIGNATURE in memory;
2. the node checks the signature of the control message;
3. if the timestamp is fresh and the signature is valid, the control message is accepted and processed according to the standard OLSR specifications for the message type. If not, both the control message and SIGNATURE message are dropped.

To fit the secured infrastructure, some modifications also need to be made to the packet processing algorithm described in the standard specifications [31]. We briefly describe these modifications. A receiving node must process an incoming packet following this algorithm:

1. if the packet contains no messages, silently drop the packet;
   (As in standard OLSR)

2. if the TTL of the message is ≤ 0, or if the message was sent by you, silently drop the packet;
   (As in standard OLSR)

3. processing condition:

   (a) if there exists a tuple in the Duplicate Set where $D\_addr =$ `Originator Address` and $D\_seq\_num =$ `Message Sequence Number` and $D\_timestamp =$ `Timestamp`, then do not process this message because it has already been processed;

   (b) else process the message according to its Message Type.
   If the message is a SIGNATURE, then

      i. if the timestamp (from the `Timestamp` field) is fresh, then maintain the SIGNATURE message (with its header) in memory. Otherwise, drop the message and erase its Duplicate Tuple from the Duplicate Set;

   Else if the message is of another Message Type that you implement, then

      i. if the `Message Sequence Number` of the message = `Message Sequence Number` of the SIGNATURE in memory +1, then continue. Otherwise, drop the message and erase its Duplicate Tuple from the Duplicate Set;
      (This step is optional)

      ii. if the computed signature (from the `Signature` field) is valid, then flush the SIGNATURE message from memory, and process the message according to the standard OLSR specifications. Otherwise, drop the message and erase its Duplicate Tuple from the Duplicate Set;

4. forwarding condition:

   (a) if there exists a tuple in the Duplicate Set where $D\_addr =$ `Originator Address` and $D\_seq\_num =$ `Message Sequence Number` and $D\_timestamp =$ `Timestamp` and the receiving interface address is listed in D_iface_list, then do not retransmit this message because it has already been considered for forwarding;

			SIGNATURE
Incorrect traffic generation	Incorrect HELLO generation	ID spoofing	✔
		link spoofing	✔
	Incorrect TC generation	ID spoofing	✔
		link spoofing	✔
	Incorrect MID/HNA generation		✔
	ANSN attack		✔
Incorrect traffic relaying	Message tampering		✔
	Blackhole attack		✔
	Replay attack		✔
	Wormhole attack		
	MPR attack		
Message bombing and other DoS			

Table 5.3: Protection offered from different OLSR attacks in absence of compromised nodes.

(b) else forward the message according to its Message Type, or to the standard forwarding algorithm if you do not implement its Message Type.
(As in standard OLSR)

Erasing the Duplicate Tuple purporting to bad messages (i.e. with a stale timestamp or an invalid signature) ensures that only good messages in the Duplicate Set are kept track of. This to avoid a DoS attack carried out by a malicious node that floods the network with junk messages not coupled to a signature message (or coupled to an invalid signature message). These junk messages fill the Duplicate Set of receiving nodes, therefore causing receiving nodes reject valid messages that bear the same MSN as a previously received junk message.

5.3 Resilience

Adding a digital signature to all control messages guarantees message authentication or integrity, as unsigned control messages coming from alien nodes are discarded. Table 5.3 shows the resilience of this security architecture to attacks, provided that any node owning a key respects the protocol (i.e. there are no compromised nodes; for a discussion on compromission of nodes, please refer to Chapter 8).

5.4 Overhead

Here we evaluate the transmission overhead of this signature protocol, compared to the standard OLSR. To give an example, we use two cryptographic schemes: a symmetric algorithm, HMAC-MD5, which results in a 128-bit digest; and an asymmetric algorithm, DSA, which results in a 320-bit signature. We do not take into account the computation overhead, i.e. the time expended in signature generation and verification, as they are machine-dependent. The computation speed is evaluated in Section 6.2.1.

5.4.1 Message sizes for the standard OLSR

The size of a HELLO message varies depending on the number of advertised neighbor nodes and on their link/neighbor status. This is because neighbors of the same link/neighbor status are listed under the same group of Neighbor Type and Link Type (identified by the `Link Code` field), and 32 bits are added for each new advertised group. There are 11 valid values for the `Link Code` field as combinations of Neighbor Type and Link Type.

Therefore, the length of a HELLO message varies greatly depending on network density, neighbors' distance, and nodes' speed. This makes difficult choosing a sample. For instance, we may speculate that, amongst n advertised neighbors, the number of different link/neighbor status is half of the number of advertised nodes, obtaining the following function for the "common" HELLO size: $32 + 48n$ bits.

We observe that the size of a HELLO message advertising n neighbor nodes is bounded by the following limits:

- minimum HELLO: $64 + 32n$ bits
- maximum HELLO: $32 + 64n$ bits if $n \leq 11$ $\quad$ $384 + 32n$ bits if $n > 11$

From these numbers we can compute (as an arithmetic mean) the average size of a HELLO:

- average HELLO: $48 + 48n$ bits if $n \leq 11$ $\quad$ $224 + 32n$ bits if $n > 11$

The average OLSR neighborhood counting from 9 to 12 nodes, we can re-average the results to obtain a linear function. We are conscious that this gives a roughly approximated value, however it is sufficient to give an idea of the message size. (This value coincides with the "common" HELLO function for $n = 13$.)

We obtain the following result:

HELLO: $136 + 40n$ bits

The size of a TC message advertising n neighbor nodes is:

TC: $32 + 32n$ bits

These are the sizes of each control message, without the message header. Considering also the IP header (160 bits), the UDP header[3] (64 bits), and the OLSR packet header (32 bits + 96 bits per message), the resulting packet lengths including all headers are:

HELLO (packet): $488 + 40n$ bits
TC (packet): $384 + 32n$ bits

These are the sizes of a packet containing only one HELLO or TC. We assume that the IP datagram is not fragmented, and that node addresses are in IPv4 format.

5.4.2 Message sizes for OLSR with signatures

The SIGNATURE message being 32 bits in size plus the size of the `Signature` field, the sizes of a packet containing a signed HELLO/TC message are:

HELLO + SIGNATURE HMAC-MD5 (packet): $744 + 40n$ bits
HELLO + SIGNATURE DSA (packet): $936 + 40n$ bits
TC + SIGNATURE HMAC-MD5 (packet): $640 + 32n$ bits
TC + SIGNATURE DSA (packet): $832 + 32n$ bits

We assume that each HELLO/TC message and its companion SIGNATURE message are sent together in the same OLSR packet, and that the packet does not contain other messages. This is a "worst case" scenario, as including more control messages in the same packet, along with the signatures of these messages, would reduce the overhead.

Figure 5.3 and Figure 5.4 show the diagrams comparing the packet overhead, full headers included, for unsecured and secured HELLO/TC messages. The figures compare the size (drawn as a line for better readability) of a packet containing a HELLO/TC with the size of a packet containing a HELLO/TC plus its SIGNATURE.

5.4.3 Flowrates

An estimation of a node's flowrate, for both the standard OLSR and OLSR with signatures, gives the following figures:

Standard OLSR: 558 bit/sec
OLSR with HMAC-MD5 SIGNATURE: 738 bit/sec
OLSR with DSA SIGNATURE: 872 bit/sec

We utilize as model a node advertising 9 neighbors (an average neighborhood size) in its HELLO/TCs. The node broadcasts a HELLO every 2 seconds, and a TC every 5 seconds. The model assumes that the node has one interface, so

[3] OLSR packets are communicated using UDP, port 698.

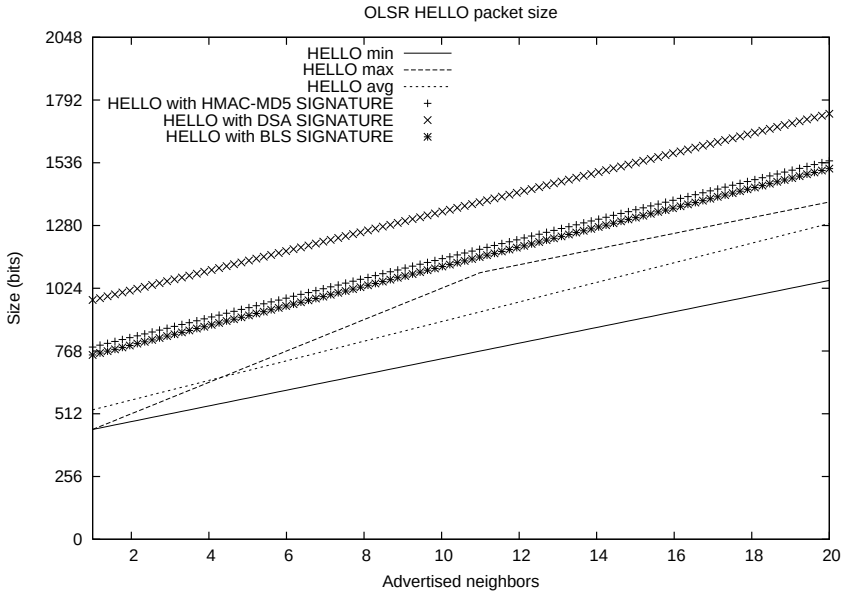

Figure 5.3: Diagram of HELLO message overhead.

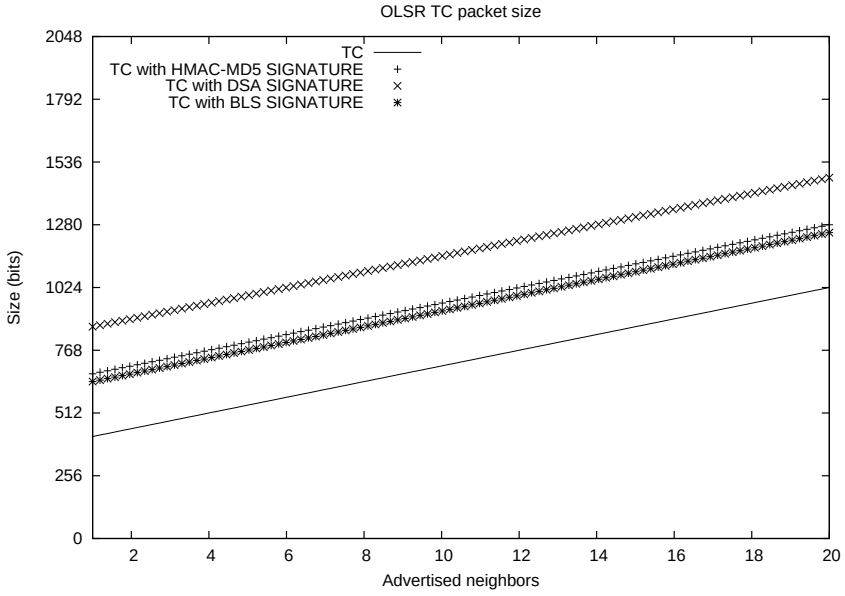

Figure 5.4: Diagram of TC message overhead.

Protocol	**Key type**	**HELLO size (bits)**
Standard OLSR	–	848
OLSR with SIGNATURE	HMAC-MD5 DSA	1004 1296
Secure OLSR [60]	HMAC-SHA1	784
MAE [128]	512-bit RSA Certificate 2048-bit RSA Certificate	1192 min, 5176 max 2728 min, 6712 max

Table 5.4: Comparison of message overhead for standard and secured OLSR.

that MID and HNA messages are not emitted. Each OLSR packet contains one HELLO or TC; plus, in the secured version, its associated SIGNATURE message. These values include the computation of IP, UDP, and OLSR packet headers, with all the assumptions made above.

5.4.4 Comparison with other solutions

We analyse here the overhead for a whole OLSR packet containing a HELLO, comparing the overhead of OLSR with SIGNATURE with that of other security solutions: Secure OLSR [60] and MAE [128]. Results are shown in Table 5.4. We assume an average neighborhood of 9 nodes. The results for MAE concern messages not including CERT objects. A quantity of 352 bits has been added to the figures regarding MAE to include the IP, UDP, and OLSR packet headers; these figures are given for a network from 10 (min) to 1000 (max) nodes [127].

Chapter 6

Cryptosystems for the ad hoc environment

Once that the security architecture has been designed in terms of which routing protocol to use, it is necessary to precise the requirements that a cryptographic infrastructure must satisfy in order to be usable.

As illustrated in Section 2.1, symmetric cryptography is fast and light for encryption and digesting, while asymmetric cryptography is efficient for signature and multiple key management.

Asymmetric algorithms offer many advantages in the securing process of an ad hoc network. However, these ciphers are unsuitable when the nodes are unable to verify asymmetric signatures quickly enough, or when network bandwidth is insufficient.

6.1 Requirements

In a generic way it is desirable that the signature algorithm used in ad hoc networks has these characteristics:

- a short signature (in bits), to minimize message overhead;
- a fast signature verification time, to prevent an intruder perform a DoS attack just by sending a large number of false signatures;
- verification faster than signing, because a message generated and signed by one node has to be verified by several (or all) nodes in the network;
- low complexity, because of the CPU power limitation of nodes in a mobile ad hoc network.

The same applies for a hashing algorithm, with the remark that generation and verification of the digest are the same operation.

An extremely strong algorithm is usually not required; the algorithm should be strong enough only to protect the exchanged messages until the next key renewal. In this point of view, a smaller key may be suitable.

6.2 Algorithm analysis

Choosing which cryptographic scheme to use for the protection of the messages is not an easy task. The choice depends largely on the requirements: whether we want to identify messages from each node (i.e. ensure non-repudiation) or just guarantee the integrity of messages – hence if we have to use asymmetric key pairs or just a symmetric key; available techniques for key distribution; computational complexity; robustness against different kind of cryptanalysis; size of the signature or digest; required time for signature generation and verification, or digest generation; and more. Furthermore, once the requirements are set, an algorithm can be carefully implemented in software and/or dedicated hardware in a way to perform better than another. With this in mind, comparing the different known algorithms has sense only if all-purpose hardware is employed. The cipher should be chosen once the requirements are clear, and while looking both at the algorithms and the software and hardware available.

Asymmetric algorithms eligibles for use in ad hoc networks may include RSA, DSA, and ECNR. If a symmetric cipher must be used instead, a good choice would be HMAC with MD5 or SHA-1, i.e. HMAC-MD5 or HMAC-SHA1. Note that the MD5 hash function has been broken i.e. collisions have been found [36, 109, 156]; however, this does not compromise the security of HMAC-MD5.

6.2.1 Benchmarks

For informational use, we publish a list of benchmark tests on the Crypto++ 5.2.1 Library[1], a free C++ class library of cryptographic schemes. The speed results for the different ciphers are shown in Table 6.1, and signature/digest lengths of these ciphers are shown in Table 6.2.

We ran the benchmarks on the following machines: Intel i486 133 MHz, Intel Pentium III 1 GHz, and Intel Pentium 4 2.8 GHz. All benchmarks were computed on algorithms compiled with gcc 3.x.x and ran under Linux, kernel version 2.4. Columns marked with a † are relative to the optimized version of the Crypto++ Library, compiled with the gcc -O9 flag.

The following notes are excerpted from Crypto++'s documentation.

Schemes marked with the symbol ‡ use precomputation; values are looked up from a table of 16 precomputed powers of each fixed base to make the exponentiation operation faster.

[1] `http://www.cryptopp.com`

Operation	**486**	**486†**	**P3**	**P3†**	**P4**	**P4†**
RSA 1024 sig	570.00	380.00	13.51	8.62	7.30	5.13
RSA 1024 ver	27.78	12.50	0.66	0.28	0.32	0.16
DSA 1024 sig	212.00	168.33	4.81	3.72	1.63	2.46
DSA 1024 sig ‡	120.00	75.71	3.01	1.68	1.08	1.05
DSA 1024 ver	230.00	191.67	5.26	4.13	2.16	2.86
DSA 1024 ver ‡	193.33	113.33	4.57	2.81	1.81	1.65
ECNR GF(p) 168 sig	386.67	228.00	9.52	5.65	3.57	2.77
ECNR GF(p) 168 sig ‡	210.00	108.00	5.75	2.96	2.11	1.39
ECNR GF(p) 168 ver	755.00	436.67	20.83	11.76	7.46	5.41
ECNR GF(p) 168 ver ‡	356.67	186.67	9.35	5.05	3.69	2.44
ECNR GF(2^n) 155 sig	1170.00	255.00	35.71	9.35	11.49	4.57
ECNR GF(2^n) 155 sig ‡	356.67	92.73	10.75	2.98	3.44	1.47
ECNR GF(2^n) 155 ver	1470.00	322.50	45.22	11.76	14.49	5.81
ECNR GF(2^n) 155 ver ‡	620.00	162.86	19.61	5.26	6.06	2.51
HMAC-MD5 HELLO	$6.15 \cdot 10^{-2}$	$1.28 \cdot 10^{-2}$	$0.19 \cdot 10^{-2}$	$0.07 \cdot 10^{-2}$	$0.03 \cdot 10^{-2}$	$0.03 \cdot 10^{-2}$
HMAC-MD5 TC	$4.58 \cdot 10^{-2}$	$0.95 \cdot 10^{-2}$	$0.14 \cdot 10^{-2}$	$0.05 \cdot 10^{-2}$	$0.02 \cdot 10^{-2}$	$0.02 \cdot 10^{-2}$

Table 6.1: Benchmarks for different ciphers (msec/op).

Algorithm	**Signature**
RSA 1024	1024
DSA 1024	320
ECNR GF(p) 168	336
ECNR GF(2^n) 155	310
HMAC-MD5	128

Table 6.2: Signature length of different ciphers (bit).

The implementations of RSA and ECNR follow the IEEE P1363 [76, 77] standard. RSA uses 17 as the public exponent, while DSA uses a 160-bit long value for q. ECNR is done over the Galois field GF(p) and GF(2^n): operations in GF(2^n) are accomplished using trinomial basis and, compared to the other algorithms, Crypto++'s implementation of ECNR over GF(2^n) is less optimized.

The generation of a HMAC-MD5 hashing is done over an average HELLO and TC signed message advertising 9 neighbors, as reported in Section 5.4.

6.3 Key management

With reference to the problematics explained in the previous chapters, asymmetric cryptography appears to be an efficient model under many aspects. However, applying this model to an ad hoc network raises several questions and difficulties. This because the deployment of a Public Key Infrastructure includes the creation of a system for key distribution, which is often incarnated under the form of a centralized Certification Authority. However, the dependence by a centralized authority does not match very well the architecture and philosophy of an an ad hoc network, where all nodes are independent and mobile. For instance, it is highly unlikely that any node is able to connect to the CA at any time. Furthermore, a centralized entity raises a problem concerning network weaknesses; this constitutes in fact a vulnerable point, which opens the door to Denial of Service attacks or compromission of the entire network. This is a problem in itself, and even in a wired network the solution is not trivial. The state of the art includes several solutions that have been proposed for key management, as an alternative to a centralized TTP.

6.3.1 Threshold cryptography

The burden of a Certification Authority may be shared amongst many parties by using *threshold cryptography* [143, 90]. Very first threshold schemes have been studied by Shamir [141].

A (n, z) threshold cryptography scheme (with $n \geq z$) allows n parties to share the ability to perform a cryptographic operation, such as a digital signature, which can be done jointly by any z parties, where the same operation is infeasible for a group of $z - 1$ or less parties. For a network of n or more nodes, the CA's secret key is divided into n shares, and each share is assigned to a node of the network. Each of the n nodes then compute a partial signature for a certificate and submit its partial signature to a "combiner" node; after receiving z partial signatures, the combiner is able to generate a correct signature for the certificate. This scheme therefore tolerates up to $z - 1$ compromised nodes.

A share refreshing system [165] allows the nodes to regenerate new shares, protecting the network against an attacker that may compromise more than $z - 1$ nodes, one after each other, over time. Another optimization called *dynamic coalescing* [100] deals with the problem of contacting enough nodes at the same time in an ad hoc network, whose topology is by its very nature constantly mutating.

The threshold cryptography scheme has been implemented in a wired environment with the COCA (Cornell Online Certification Authority) framework [166] and in an ad hoc wireless network with MOCA (MObile Certification Authority) [162]. There exists also an implementation for the OLSR framework [34].

6.3.2 Self-organized PKI

Čapkun et al. propose a public-key management system [151, 69] in which no CA exists: certificates are issued by the users themselves, which build and maintain a local certificate repository. For A to verify the authenticity of B's public key, A must try to find a certificate chain from A to B in the repository build up from the merging of A's and B's local repositories. This infrastructure is based upon the analysis [152] of the PGP [167] certificate graph (*web of trust*). The PGP trust graph exhibits the *small-world* property, i.e. it has a small diameter and is highly clustered.

6.3.3 Identity-based cryptosystems

Identity-based encryption (IBE) [47] is a form of public key cryptography in which the public key of any participant is derived from the identity, or any other intrinsic quality, of the participant itself. For instance, the public key of a node can be its IP address. In this way there is no need for a CA, as a public key is bound unambiguously to a specific participant. Identity-based encryption has also other interesting properties, such as simplifying key revocation, key delegation, and user credentials management.

Identity-based encryption requires nonetheless the presence of a Trusted Third Party, called Private Key Generator (PKG), which firstly generates the master key. A node communicates via a secure channel with the PKG, requesting the private key corresponding to its identity – its IP address in the previous example. The node can afterwhile use its private key to decrypt messages sent to it.

Shamir was the first to concoct identity-based cryptosystems [142]. After him, Boneh and Franklin [16] designed an efficient and secure IBE scheme, which was further ameliorated by Lynn [101] with the addition of message authentication. Lynn's scheme guarantees that the integrity of the message is preserved, serving as a digital signature scheme.

6.3.4 Imprinting

Perhaps the easiest solution at all is to require that a node accepts a key only at the bootstrap, possibly manually (by direct contact). This is called *imprinting* by Stajano and Anderson [144], with reference to ethology: imprinting is the phenomenon which makes a duckling emerging from the egg choose the first seen animated object as mother. Another system for key renewal can subsequently prevent keys becoming stale. This solution is used by Balfanz et al. to provide pre-authentication on a location-limited wireless channel [7].

6.3.5 Probabilistic key distribution

Some schemes relies on probabilistic key distribution methods on a distributed environment to establish pairwise keys [41, 97]. In these schemes, each node picks up randomly a certain number of keys from a key pool, so that any pair of nodes has a certain probability to share at least one same key.

6.3.6 Diffie-Hellman key agreement

In a key agreement protocol, two (or more) parties derive a shared secret key from information contributed by each party and exchanged over a channel that does not need to be secure. Eavesdropped information does not lead to disclosure of the secret key.

The Diffie-Hellman key agreement protocol [35] is the first public key algorithm invented. Its security comes from the fact that computing exponentiation in a finite field is easy, while the inverse operation of calculating discrete logarithms in unfeasible. The original DH key exchange protocol is designed for two parties, but it can be extended to three or more parties (generalized Diffie-Hellman). This extension leads to the Group Key Agreement protocols [146, 12, 6].

The generalized DH protocol may therefore be employed to establish keys in an ad hoc environment; it is used for instance in SRP. The CLIQUES family of protocols [147, 161] for authenticated group key distribution is based on Diffie-Hellman. In the simplest form of the CLIQUES protocol, the computation of the key proceeds from node to node, the last node broadcasting the result to allow the other nodes to generate the final key.

6.3.7 A simple PKI for OLSR

We outline two simple PKIs for OLSR. They both serve the purpose of making public keys available to nodes in the network in a way such that the authenticity of the keys can be trusted. The two PKIs differ mainly in that the first is proactive, in the way it aims at diffusing periodically public

key information to nodes in the network, while the second is reactive: nodes request keys only when needed.

Proactive PKI for OLSR

This PKI operates with three classes of nodes:

Untrusted nodes: A node A considers another node X as an *untrusted node* if the public key of X is not known by A, or if this public key is known but not validated by a signing authority in the network. That is, messages' signatures received from an untrusted node cannot be verified. Note that at network initialization all nodes, except the signing authority and the node itself, are untrusted from the point of view of the individual nodes.

Trusted nodes: A node A considers another node X as a *trusted node* if the public key of X is known by A and this public key has been validated by a signing authority in the network. That is, signatures of messages received from trusted nodes can be verified.

Signing authorities: A *signing authority* is a node which has the special property that its public key is a priori known by all nodes in the network. A signing authority has special responsibilities for the network, namely to allow new nodes to register their public keys in a secure fashion (typically through manual authentication), whereby a new node becomes a trusted node; and to periodically distribute signed certificates, containing a list of public keys for all trusted nodes.

As an option, it is also possible to use the PKI infrastructure for time synchronization, and have the signing authority periodically distribute the signed time.

Each node that wishes to participate in the network is required to register its public key with a signing authority. The signing authority will issue certificates periodically, which will then be broadcast to the entire network. Nodes receiving the certificates will store these for a specified amount of time, after which they expire. Hence periodic refresh of certificates is required.

No explicit mechanisms for revoking keys is presented. To facilitate key revocation, certificate messages may be equipped with a sequence number associated with the set of keys advertised. Whenever the set changes (when keys are added or removed) the sequence number is incremented, and included in following certificate messages. Upon receiving a certificate message the nodes can distinguish between older and newer information, and remove expired keys. In order to counter possible replay attacks, timestamps should be employed.

While we could foresee to use this architecture to reject messages from untrusted nodes, this is an approach that must be carefully reviewed, as it is proven that it leads to a deadlock at network initialization. In fact, at the bootstrap, before the signing authority start distributing certificates, all nodes are untrusted; if their control messages are rejected, then formation of the network will never take place, and without network, the certificates cannot be spread to nodes. Next we present a detailed discussion of the problem and of the proposed solution.

Admittance control

From the perspective of network connectivity, the primary aim is to ensure that false topology information is not spread amongst the nodes. This translates into protecting the integrity of OLSR's most important feature: the creation and relaying of TC messages through MPRs. Therefore, a node should:

- select only trusted neighbors as its MPRs;
- accept to be selected as MPR by trusted neighbors only;
- accept TC messages originating from trusted nodes only[2];
- forward broadcast messages from trusted neighbors only.

When node A selects node B as MPR, A gives to B the responsibility for the $A-B$ link. This responsibility is fulfilled only if B is trusted; otherwise, there is an hazard of B performing incorrect traffic relaying, as described in Section 3.2.

When node A is selected as MPR by node B, A assumes the responsibility for the contents of TC messages coming from B. If B is not trusted, the hazard is that it could inject false TC messages in the network. This is an instance of incorrect traffic generation. The same happens when node A accepts TC messages originating from node B, except that in this case the hazard is circumscribed to A only.

The situation is similar when node A forwards broadcast messages from node B. If B is not trusted, it could maliciously generate excessive amounts of broadcast traffic that, once flooded to the network, may consume excessive resources and potentially prevent transmission of legitimate traffic.

The simplest possible mechanism to keep untrusted nodes out of the network would be a rule stating: "A message sent by an untrusted node is silently discarded and neither processed nor forwarded." However, while simple, this condition is too restrictive and not applicable. If all nodes require that all traffic they receive must be signed and verified, accepting therefore

[2]These trusted nodes are MPRs of other nodes in the network, because we specified already the condition that the node should select only trusted neighbors as its MPRs.

only traffic from trusted nodes, this would lead to a deadlock problem on network initialization, when public key certificates start being spread around.

In fact, upon network initialization, no node know any public keys other than that of the signing authority (and, of course, their own). Disregarding control traffic from the signing authority, all nodes will by default ignore control traffic from each other. Thus, no nodes will select MPRs, and no broadcast messages will be forwarded. The only node which may be selected as MPR is the signing authority itself. Control traffic from the signing authority will be accepted by its neighbors since they know the public key of the signing authority in advance; until the signing authority starts broadcasting certificate messages, no network formation will take place. Unless special provisions are made, only neighbor nodes of the signing authority will ever receive the broadcast certificates: since successful verification of a signature is a criteria for accepting any control messages, 2-hop neighbors of the signing authority will never accept control messages from 1-hop neighbors of the signing authorities. This implies that a symmetric link between 1-hop and 2-hop neighbors of the signing authority will never be established. The signing authority will therefore never select MPRs and, subsequently, its certificates will never be broadcast into the network.

To avoid this situation and enable network initialization, special provisions for accepting some control messages without validation of signatures must be made. The desired goal is to allow MPR flooding to take into account the fact that broadcast messages should be able to reach also untrusted nodes in the network. Hence the following additional conditions apply:

- A node must accept unsigned HELLOs from untrusted neighbors. Such HELLO messages are accepted under the restriction that:
  - asymmetric and symmetric links are considered as such;
  - MPR links are considered as symmetric only (i.e. they do not affect the MPR selector set);
  - lost links are ignored;
- A node must maintain a *trusted neighborhood* containing information about links to the trusted nodes in its neighborhood;
- A node must maintain an *untrusted neighborhood* containing information about links to the untrusted nodes in its neighborhood;
- A node must, from among the trusted neighbors, perform MPR selection as specified;
- A node must periodically transmit HELLO messages, including the trusted neighbors (with status: asymmetric, symmetric and MPR, as appropriate) and untrusted neighbors (with status: asymmetric and symmetric only).

The 2-hop neighborhood of a node will contain both trusted and untrusted nodes. MPRs are selected from among the trusted nodes such that, as much as possible of, all nodes in the 2-hop neighborhood are covered. This ensures that a maximum of untrusted neighbors of trusted nodes will be reached by MPR flooding, as they are covered by at least one MPR.

Thus, upon network initialization, the signing authority will transmit its certificate which will be received by its 1-hop neighbors. Following HELLO message exchange, the 1-hop neighbors will accept the untrusted 2-hop neighbors as symmetric but not select MPRs among them. The signing authority will then select MPRs from among the 1-hop neighborhood such that the next broadcast certificate will reach the 2-hop neighbors. The scheme follows, in such a way that certificates will, upon network initialization, propagate from the signing authorities and towards the edges of the network.

Note that some information coming from untrusted nodes is only used to handle untrusted nodes. MPR selection, etc. is performed only among trusted nodes, as MPR selection information is diffused only about trusted nodes.

Reactive PKI for OLSR

We assume the same framework as in the proactive PKI. Two new types of messages are introduced: Key Request and Key Reply.

A Key Request is a message from a node A, containing a nonce N_A initialized with random values for each request and a list of nodes B_i for which the public key is needed: $A \rightarrow all : \{A, N_A, B_1, \ldots, B_n\}_A$.

Upon receiving such a request message, a signing authority U:

1. first checks the signature of the message A, if it has the public key of A;
2. if the public key of at least one B_i in the request is known by the authority, a Key Reply is generated. The reply includes all the public keys it knows: $U \rightarrow all : \{U, A, N_A, (B_1, P_{B_1}), \ldots, (B_m, P_{B_m})\}_U$.

Upon receiving such a reply message, the originating node A performs the following checks:

- that the destination of the message is indeed A;
- that U is a signing authority that it trusts;
- that the signature of the message is correct;
- that the nonce N_A was a nonce it recently used.

If those checks succeed, node A finally updates its public key database with the newly acquired keys.

In order to ensure proper delivery of Key Request and Key Reply messages, pure flooding is used instead of the standard MPR forwarding. As with the proactive PKI, considerations regarding key revocation are not presented. These features, however, can be fashioned through lifetime of the public keys and periodic refreshing through renewed request-reply cycles.

Chapter 7

Timestamps

As already said, a common problem in distributed systems is that, even assuming a digest or signature is checked (therefore ensuring integrity or authentication of the source of a message), replay of previously transmitted messages is possible by an intruder. This is the aforementioned replay attack, which may easily corrupt the route cache and therefore discompose the correct functioning of the network.

Timestamps are a commonly used means to prevent replay attacks (as in Kerberos [145]), and are indeed necessary [37, 33]. The idea is to device a proof of freshness, such that older pieces of information can be detected and rejected. Further timestamp methods have been discussed by Gong [56]. When a timestamp is not sufficient, the goal is achieved by using a *nonce* [112], which is a small sequence of randomly generated bits, used only once. The nonce is sent in a challenge as an identifier and must be included in the response.

In OLSR, MSN (Message Sequence Number) and ANSN (Advertised Neighbor Sequence Number) are already used for achieving those goals in the context of allowing the routing protocol to determine which information is more recent. However while these sequence numbers are sufficient for the basic routing protocol functioning, they are not sufficient to provide full security: each are encoded on a 16-bit field, which implies that wraparound happens too frequently to provide efficient protection against malice from an intruder.

Here we describe several timestamp algorithms, providing different levels of security at the expense of different costs. For the purpose of the discussion we use the following terminology:

- a *clock* is the device, hardware or software, within a node keeping track of the time;
- a *timestamp* is the value of a clock, recorded in a piece of information (e.g. a message) at the time of generation of the information.

Commonly, the following methods are employed for providing timestamps:

- Real time: a clock expresses time in some natural resolution such as seconds or microseconds;
- Logical time: a clock is incremented each time an event occurs, such as message generation.

Note that these are just different ways to express the same idea of time: the basic property being that time is monotonically increasing, and that upon receiving a message containing a timestamp the receiving node has some idea about the value in the timestamp compared to the value of the clock, i.e. what the timestamp should be. As concerns the second option, the concept of event ordering in a distributed system has been examined by Lamport [93].

For each message being emitted by a node, an unique timestamp T_{sender} is included. Let t_0 denote the value of the clock in a receiving node around which the timestamp T_{sender} in a received message is expected to be. Then, a more formal expression of a message being "not too old" may be:

$$|T_{sender} - t_0| < \Delta t_{max} \tag{7.1}$$

where Δt_{max} is a constant used to limit the timestamp discrepancy while allowing for some small deviation. Thus, the (7.1) provides a simple framework for checking if a received message is original or rather it is a replay of a previous message.

The replay check in the (7.1) can be complemented by maintaining a *Signature Table*, in order to also prevent replays within a small time-scale i.e. replays within a delay less than Δt_{max}. The Signature Table contains the signatures (or the digests) of the most recently received messages, for a duration greater or equal to Δt_{max}. If the signature of a received message is already in the Signature Table, it is ignored since the message has already been received and processed. This is similar to the Duplicate Set in OLSR, which ensures that TC messages are processed and forwarded once. Indeed, the functionalities of both the Signature Table and the OLSR Duplicate Set could be merged.

The way in which timestamps are generated is not necessarily obvious, as they assume either synchronous real-time timestamps, non-volatile timestamps, or they implicitly require a challenge-response protocol.

It should be noted that non-synchronization leaves the door open to clock attacks. If the sender's clock is ahead of that of a receiving node, an attacker may suppress the postdated message and replay it later, when the timestamp in the message becomes valid according to the receiver's clock. Re-synchronization of the sender's faulty clock does not parry this *suppress-replay attack* [55].

In the following we present different methods concerning the use of timestamps. These methods introduce different levels of complexity, cost, and security tradeoffs; they can also be used in combination, in order to provide greater resilience.

7.1 No timestamps

If no replay protection is desired, nodes may just set the timestamp to be 0 when generating messages, and not check the timestamps upon receiving.

7.2 Real-time timestamps

A conceptually simple way to generate timestamps, although not the easiest one to implement, is to use a real time clock in each node, assuming some kind of synchronization. This solution can be achieved by having a safe source of time in each node that is sufficiently precise and with sufficiently little drift. This could be in form of a quartz clock, an atomic clock, or access to the time as obtained by a GPS device.

The criterion for accepting a message is indeed simply, as in Formula 7.1, $|T_{sender} - T| < \Delta t_{max}$, where T is the value of the clock on the receiving node.

The source of time for timestamp generation can often be a quartz clock, or similar equipment, embedded in the node's hardware. The main issue when using an internal source of time is the precision of the clock. In most computer equipment, piezoelectric quartz oscillators are used to keep track of the time; in personal computers, it is the so-called BIOS clock. The precision of these clocks is nonetheless limited causing a drift, in absolute terms, in the order of magnitude of one second per day. The drift is due to two factors: the lack of precision of the quartz, assumed to oscillate at a certain determined frequency different from the real frequency, and variations in the real frequency due to temperature, aging, vibration-induced noise, and other factors [155].

In order to assess the frequency needed for a possible resynchronization, we conducted experiments [4] with the routers used in the OLSR signature implementation described in Chapter 5. During the experiment, four routers broadcasted their BIOS clock periodically on an Ethernet network. A machine recorded the arrival times of the different broadcasts of the clocks, along with their advertised time value.

All clocks were synchronized at the beginning of the experiment. One of the routers was used as a reference, and the difference between the values of the clocks was recorded and plotted as a function. The resulting maximum clock drift is around 1 sec/day (Figure 7.1), quite comparable with expected values. However, the resulting plotted functions are mostly linear, confirming

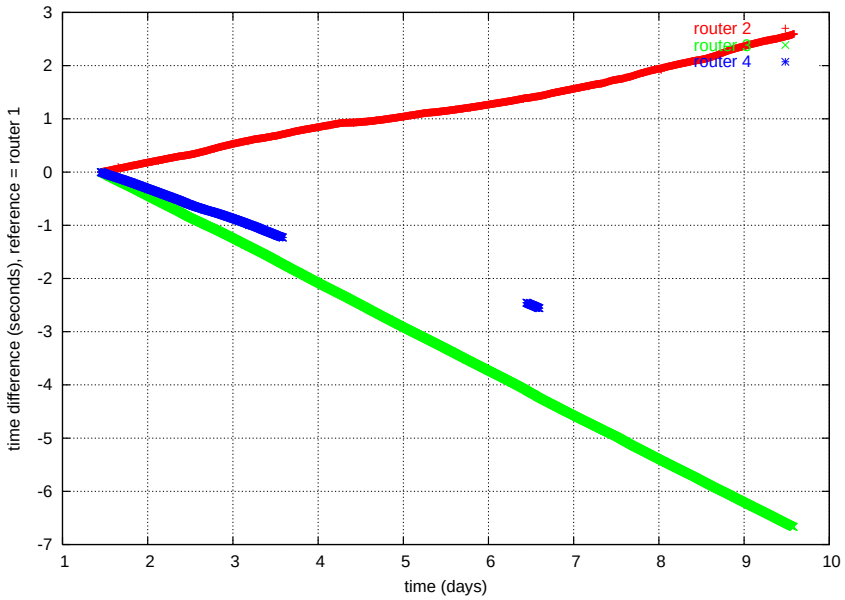

Figure 7.1: Time difference between clocks.

that the main factor in clock drift is the incorrect calibration of the real frequency of the quartz. By using linear regression on the values found, the linear component of drift between one router and the reference router was removed. In practice such corrections could be performed by making precise time difference measurements at two separate points of time. As a result, the precision is much better and around 30 msec/day (Figure 7.2). Nonetheless, the time functions of some routers were irregular; by comparing the drift estimates based on measurements between different points of time, to the drift estimates based on measurements between the points of time with the greatest discrepancy, the drift is found to be about 0.2 sec/day. The equivalent necessary synchronization intervals for drift correction, for a drift of maximum 15 seconds, are therefore 500 days in the average case, and 75 days in the worst case.

7.3 Non-volatile timestamps

A way to provide weak timestamps is to have the clock of each node of the network maintained in non-volatile memory, initialized the first time a node's signature key is used after generation.

The value of this clock is then used as timestamp in each message signed, after which the value is incremented. While the sender maintains the clock in non-volatile memory, the receivers maintain a table containing the maximal

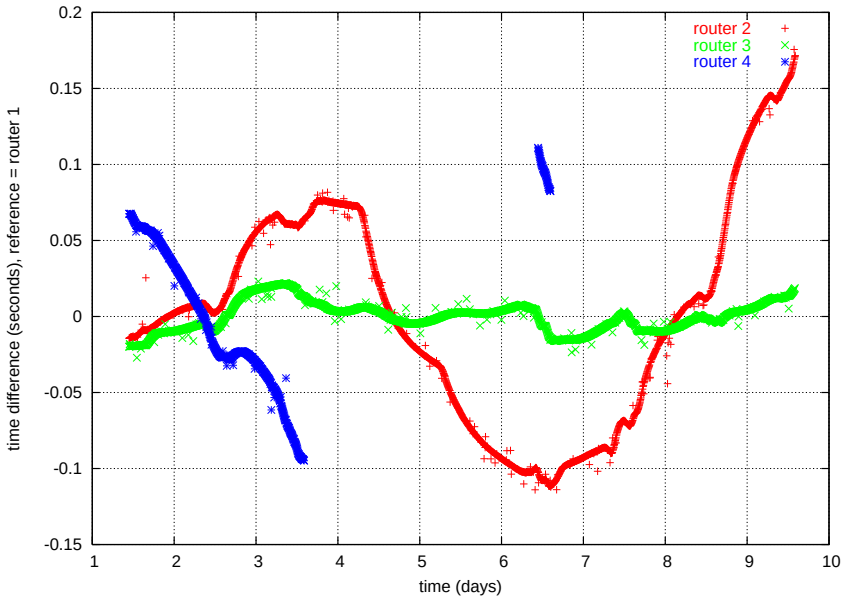

Figure 7.2: Time difference between clocks, after resynchronization.

timestamps received from all nodes in the network.

In the receiver node R, the algorithm for processing a message from sender S with timestamp T_S is the following:

1. R keeps the value of T_S^R, the highest timestamp from S ever received, in non-volatile memory;

2. if R has not received any value from S, R considers the highest timestamp received from S to be $T_S^R \leftarrow 0$;

3. if $T_S > T_S^R - D$ then the message is accepted, otherwise it is rejected and not processed further. D is some fixed (small) constant to allow for out of order reception of messages. D must be tuned accordingly to the specifications of the ad hoc network;

4. T_S^R is updated: $T_{S,new}^R \leftarrow max(T_S, T_S^R)$.

These are security-wise weaker timestamps, since if communication between the sender and the receiver is broken for some time, then the latter goes out of sync and all the messages from the sender can be replayed to the receiver. This is especially true if the sender and the receiver are in different, non-connected, networks. The advantage is that as soon as the receiver and the sender are able to communicate with each other, only limited replay is possible. This replay can further be suppressed with the Signature Table, described previously.

In OLSR, non-neighbor nodes may never exchange messages, because nodes which are not selected as MPR by any other nodes exchange messages only with their neighbor nodes. In this case, the timestamps would never be updated. Therefore it would be necessary that each node periodically broadcasts at least an empty message in order to provide synchronization.

Note that a variant using a local "wall clock" time instead of incremented timestamps is possible, and could allow more stringent checks, although the algorithm still remains vulnerable.

7.4 Clock synchronization

With respect to clock synchronization, the chicken-and-egg dilemma arises [56]: timestamps are used for authentication, but secure clock synchronization itself requires authentication. This circular dependency is overcome by performing authentication and clock synchronization operations at the same instant. However the main problem with secure clock synchronization in an ad hoc network is that many algorithms require a fixed percentage of non-compromised nodes in order to operate. It could be argued that, since in an ad hoc network an intruding node can impersonate as many fictional nodes as it wishes [39], under the limitation that these fictional nodes have keys known to the network, a guaranteed fraction of non-compromised nodes is unobtainable. Even a clock synchronization algorithm such as that proposed by Dolev et al. [37], which does not require any such fraction of correct nodes to run properly, provenly cannot bound the necessary delay of synchronization when a new node wishes to participate in the network for the first time. This is quite problematic in a wireless ad hoc network, since nodes are expected to be able to leave and join at any time.

7.4.1 Timestamp exchange protocol

This part describes a timestamp exchange protocol that can be applied to OLSR. It essentially mixes a distributed challenge-response protocol with timestamp information. This protocol is a variation over the Needham-Schroeder public key protocol [112], albeit with a superset of the information (and includes, for instance, the necessary correction of the protocol proposed by Lowe [99]), using signatures instead of encryption, and using timestamps instead of nonces.

The assumption for the protocol is that each node X keeps a clock T_X, whose value is used in the timestamp fields of generated messages. The clock increases monotonically with each message sent, and with wall clock. At a given wall clock time t, the clock in the node A is denoted $T_A(t)$. The clock T_A is also used as a nonce, and thus should be initialized, fully or in part, with random values.

A simplified version of the protocol is given first, illustrating the gist of the protocol limited to two nodes A and B.

1. At t_0, $A \rightarrow B : \{A, T_A(t_0)\}_A$;

2. At $t_1 > t_0$, B has already received the previous message and sends its message: $B \rightarrow A : \{B, T_B(t_1), A, T_A(t_0)\}_B$;

3. At $t_2 > t_1$, A has received the previous message and sends its message: $A \rightarrow B : \{A, T_A(t_2), B, T_B(t_1)\}_A$.

The idea is that at t_2 (step 3) A had received the message sent in step 2, and thus observed that a recent version of its timestamp $T_A(t_0)$ was included in a message, authenticated to be from B. Therefore A can safely assume that $T_B(t_1)$ is a recent value of the timestamp from B, posterior to t_0. This relies on A properly generating initial values of clock T_A i.e. not (or with low probability) repeating values over the time.

Likewise, upon receiving the message sent from A to B in step 3, B concludes, like A, that it has now a recent authenticated value of T_A. After those steps are complete, A and B both have knowledge about relatively recent values of each others respective timestamps, which are not (or with very low probability) the result of replays. In this case we say that the handshake is completed.

A detailed parallel version of the algorithm now is given. It is "parallel" in the sense that the same message, sent by a node A to perform the previously illustrated handshake, is sent this time to several nodes (ideally all) in the network, rather than to an individual node B. Also some provisions are taken for being able to practically perform timestamp check, and for switching to new timestamp intervals.

The protocol relies on an unique new kind of message, a Timestamp Exchange message, being flooded periodically by each node. When maximal security is desired, the message should be transmitted by pure flooding to the network instead of using the MPR forwarding optimization.

We assumed that each node keeps a table of the information from the latest Timestamp Exchange message it received from each node. This table is called the *Timestamp Table*. A node A records the following information for each other node B:

- a boolean H_B^A indicating whether the handshake with B has been completed,

- the timestamp T_B^A from the latest Timestamp Exchange message received by node A from node B (in case the handshake is completed), or the list of the latest timestamps $T_{B,j}^{*A}$ received (in case the handshake is not completed),

- a set of different timestamp interval tuples for $i = 1, \ldots, n_B^A$:

$$\langle T_{B,min,i}^A, T_{B,max,i}^A, E_{B,i}^A \rangle$$

  where $E_{B_i}^A$ is an expiration time indicating that the tuple should only be used until this time value is reached, when the handshake is completed.

In node A, each timestamp interval tuple of B describes a valid interval for timestamps of B. There are several such timestamp interval tuples for B (several is) in order to allow for timestamp interval changes. Such a change would occur, for instance, when the node decides to regenerate a clock's value from scratch. The timestamp interval tuples are used with the following timestamp check: in node A, at t, a timestamp T_B from a message from B is valid if and only if an i exists such that $T_{B,min,i}^A \leq T_B \leq T_{B,max,i}^A$ and $t < E_{B,i}^A$.

This timestamp check does not apply to Timestamp Exchange messages themselves, described below.

At node A, given two valid timestamps T_B and T'_B from B, an ordering relation can be established for comparison. Let j and j' be the indexes such that $T_{B,min,j}^A \leq T_B \leq T_{B,max,j}^A$ and $T_{B,min,j'}^A \leq T'_B \leq T_{B,max,j'}^A$. Then we decide that $T_B > T'_B$ if and only if $j > j'$ or ($j = j'$ and $T_B > T'_B$). This is used for determining which of two messages, to which the timestamps relate, is the most recent.

For protocol completeness, in node A the timestamp T_B^A is said to be *orphaned* when H_B^A is false or when T_B^A does not pass the timestamp check with any of the timestamp intervals and expiration time $\langle T_{B,min,i}^A, T_{B,max,i}^A, E_{B,i}^A \rangle$. This can occur when some (or all) of those intervals expire, usually meaning that communication between node A and node B is broken.

This yields a formal definition of "completion of handshake between A and B": a handshake is complete for node A when B has a timestamp T_B^A which is not orphaned. Each time H_B^A was true and the timestamp T_B^A becomes orphaned because of timestamp interval, H_B^A is updated as: $H_B^A \leftarrow$ false.

The protocol relies on two parts: the generation and the processing of Timestamp Exchange messages.

The algorithm for the generation of the messages is as follows: node A sends periodically a Timestamp Exchange message, containing:

- its current clock T_A^{new}, which must not be orphaned with respect to the bounds set out in the following item;

- its current timestamp bounds set n_A^{new}, and for $i = 1, \ldots, n_A^{new}$:

$$\langle T_{A,min,i}^{new}, T_{A,max,i}^{new}, D_{A,i}^{new} \rangle$$

  where $D_{A,i}^{new}$ is the maximal duration for which the tuple should be kept;

- the content of its timestamp table, tuples $\langle X, T_X^A \rangle$ for each node X from which it has received a timestamp. Note that for each X with which the handshake is not completed there might be several $\langle X, T_{X,j}^{*A} \rangle$. In this case, once the $\langle X, T_{X,j}^{*A} \rangle$ has been sent, the tuples are removed;

- the signature of this message.

Node A also records the latest timestamp bounds set it has sent: $(T_{A,min,i}, T_{A,max,i})$.

The algorithm for the processing of the Timestamp Exchange messages for a node A receiving a message from node B is as follows:

1. check for the entry of B (if it exists) in the Timestamp Table to determine if H_B^A was true but the T_B^A has become orphaned. If T_B^A has become orphaned, then $H_B^A \leftarrow$ false;

2. if no reported timestamp from A, inside the Timestamp Exchange message, pass the timestamp check in A, or if there is no T_A in the message, then: if no entry for B was recorded, or if H_B^A is false, the timestamp from the message T_B is added to the list of the timestamps $T_{B,j}^{*A}$.

   The idea here is that node B has not provided enough proof of freshness for A to accept the timestamp intervals. However T_B should be kept such that in next message from A it would serve as proof of freshness. All T_B received should be kept since some could constitute invalid replays;

3. otherwise, the handshake is certain to be completed and the timestamp bounds are updated if necessary:

   (a) if no value T_B^A was recorded or if H_B^A was false, or if the new timestamp T_B of the message is greater than the latest timestamp recorded T_B^A (with the timestamp comparison rules given previously), then:

      i. T_B^A is updated with the timestamp from the message, the previous list $T_{B,j}^{*A}$ is emptied, and $T_B^A \leftarrow T_B$;

      ii. the timestamp bounds for B are updated with the values from the Timestamp Exchange message: $n_B^A \leftarrow n_B$, and for $i = 1, \ldots, n_B$:

      $$\langle T_{B,min,i}^A, T_{B,max,i}^A, E_{B,i} \rangle \leftarrow \langle T_{B,min,i}, T_{B,max,i}, D_{B,i} + t \rangle$$

      where t is the wall clock at node A.

   (b) $H_B^A \leftarrow$ true.

It is expected that the timestamp bound set of a node is limited to an interval $(T^1_{A,min}, T^1_{A,max})$, and only occasionally updated when a new timestamp interval $(T^2_{A,min}, T^2_{A,max})$ is generated. The transition is typically the following:

1. A advertises the interval $(T^1_{A,min}, T^1_{A,max})$ and generates timestamps in this interval;

2. for some duration, A advertises interval $(T^1_{A,min}, T^1_{A,max})$ and a new interval $(T^2_{A,min}, T^2_{A,max})$. Node A will still generate timestamps from the first interval, to wait for the new interval to be updated in receiving nodes;

3. for some small duration, A advertises both interval $(T^1_{A,min}, T^1_{A,max})$ and interval $(T^2_{A,min}, T^2_{A,max})$. A now generates timestamps from the second interval, while it keeps advertising the old interval;

4. A advertises only interval $(T^2_{A,min}, T^2_{A,max})$.

Note that it is possible to add variations: updating also the timestamp table on reception of messages like HELLOs, introducing some maximum deviation Δt_{max} from the last received timestamp, or using local wall clock (possibly with a random offset) instead of an incremental counter. It is possible to add optimizations to avoid sending lists of timestamps of the same node before handshake completion, such as sending immediately a Timestamp Exchange message, along with Denial of Service detection with respect to the handshake protocol.

Chapter 8

Security in ad hoc networks: advanced mechanisms

As previously seen, using message signatures effectively protects the network against identity spoofing attacks, as long as the signature mechanism is not broken. Nodes rely on signatures to identify the real sender of a message, and, with the assumption that all nodes are well-behaving, signed topology information is assumed to be correct. The scenario is hence an ad hoc network with a deployed, working PKI and message signature mechanism.

8.1 Compromised nodes

We conjecture now that an attacker has been able to gain full control – physically, or in any other way – over a trusted node, hence the attacker has now gained a privileged position inside the network. The control messages the attacker can send will be accepted as valid by all other nodes because they are correctly signed, even if these control messages are wrong. The term *compromised node* designates such a trusted node that has been taken over by the attacker.

We extend the definition of compromised node to a node which may not be under the control of the attacker, but whose private key has been disclosed to the attacker. In some way or other, the attacker has managed to capture the node's private key, stealing the node's identity, and can send messages signed on behalf of that node.

In this scenario, any trusted node is no longer trustworthy, because it could send wrong control messages to maliciously perturb the network topology. The question is: "How can we be sure that the information from a node X is correct?" There is no thing such as an "evil bit" [11] that would allow us to distinguish good information from bad.

We can nonetheless increase the odds of distinguishing good nodes from bad ones by adding redundant information in messages, so that the detec-

			SIGNATURE	ADVSIG	SIGLOC	Accusation
Incorrect traffic generation	Incorrect HELLO generation	ID spoofing	✔	✔	✔	✔
		link spoofing		✔	✔	
	Incorrect TC generation	ID spoofing	✔	✔	✔	✔
		link spoofing		✔	✔	
	Incorrect MID/HNA generation		✔	✔	✔	✔
	ANSN attack		✔	✔	✔	✔
Incorrect traffic relaying	Message tampering		✔	✔	✔	✔
	Blackhole attack					✔
	Replay attack		✔	✔	✔	✔
	Wormhole attack				✔	
	MPR attack					✔
Message bombing and other DoS						✔

Table 8.1: Protection offered from different OLSR attacks in presence of compromised nodes.

tion of wrong messages is easier. This prevention mechanism aims to prevent nodes being compromised at the outset. We propose a solution based on multiple signatures (ADVSIG) in Chapter 9, and a solution based on geographical information of nodes (SIGLOC) in Chapter 10.

We recall that the model used is an ad hoc network where each node uses public key cryptography to authenticate messages and to preserve their integrity, hence the following solutions presume the use of asymmetric cryptographic schemes. On the other hand, when a shared secret key is used, it is much more difficult to take countermeasures, because the compromised node can masquerade as any other node in the network.

A detection mechanism that can be used in parallel with a cryptographic scheme, but does not require it, is the behavior audit of nodes, to identify misbehaviors. Nodes are monitored to check that they follow the protocol correctly; the duty of monitoring is often distributed among all the nodes. Once a misbehaving node has been detected, the other nodes (the legitimate ones) should take corrective action to prevent the misbehaving node from participating any further in the network. Behavior monitoring is discussed in Chapter 11, as well as our proposed solution for OLSR which uses broadcast of accusation messages.

Table 8.1 resumes these different security architectures, and shows which attacks are assessed by each specific solution.

Chapter 9

Using multiple signatures in OLSR

When an attacker commands a compromised node, he is able to perform link spoofing (as described in Section 3.2) with the purpose of perturbing the network. To withstand the link spoofing attack, we have designed a protocol which uses multiple signatures (generated by different nodes) to validate link state information [130]. Like the solution for OLSR signatures, our protocol relies on creating and sending a new additional message in conjunction with routing control messages. We call such a message an ADVSIG (for ADVanced SIGnature) message. Our main approach is based on authentication checks of new information injected into the network, and reuse of this information by a node to prove its link state at a later time. To our knowledge, this is the first time such a protocol has been proposed.

In general, HELLO and TC control messages have the semantics of the originator advertising "I have a link with these other neighbor nodes". The signature on these messages, introduced in Section 5, serves to verify that the originator is indeed the one claiming such a link to exist. The task is now to validate that the other nodes also believe such a link to exist.

9.1 Topology continuity

In OLSR, and in any other link state protocol, the network topology, with respect to the local neighborhood of a node, is related to what the network topology was at a previous instant. This because the link state at a given time t depends on the link state at an immediately previous time $t - \Delta t$.

E.g. at time t, node A selects node B as a MPR. We can therefore state that, at time $t' = t - \Delta t$, node B declared a symmetrical link with node A. We can further state that, at time $t'' = t' - \Delta t'$, node A had an asymmetrical link with B (i.e. A heard B), and declared this fact in a HELLO message which was received by B. In fact, this is exactly the way the nodes verify

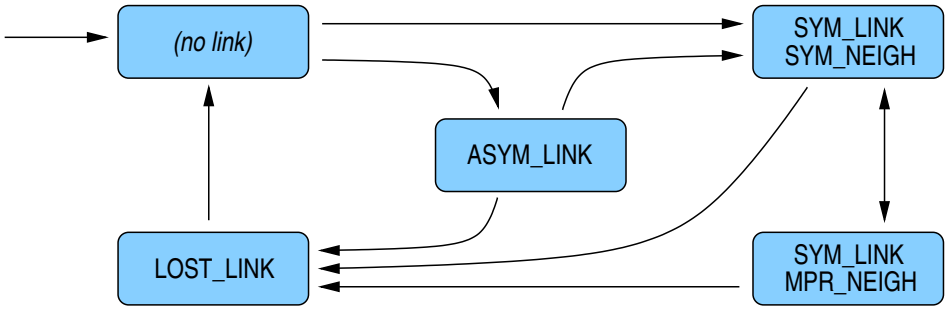

Figure 9.1: The finite state machine for OLSR link state transitions.

and establish symmetrical links in order to build a connected network; these steps, and their order in chronological sequence, are mandatory. Here we assume that all nodes correctly follow the protocol.

In summary, topology does not make leaps; instead, it proceeds smoothly with continuity. The transitions of link states is modeled in the automaton shown in Figure 9.1.

We might exploit this fact to avoid false routing information being injected in the network. The philosophy is that every node stores the most recent link state information about itself, as received by its neighbors (in their HELLOs); then the node reuses this information by including it, as a proof, in its control messages (HELLOs and TCs). In this way a node can prove that it supplies routing information accordingly and consistently with its previous neighborhood status. Of course, link state information has to be signed by the node that generated the message, otherwise a compromised node could easily produce false proofs.

9.2 Link Atomic Information

It would be inefficient to sign and redistribute a whole HELLO message as a proof, because each HELLO contains many links related to many nodes. As OLSR control messages are not modified, we should split this data into reusable pieces of information.

In order to keep the protocol as light and simple as possible, we must identify the minimal quantity of exchanged link state information. The *link atomic information* generated by a node A concerning a neighbor node B consists of:

- the address of A as the originator node
- the address of B as the advertised node

Link to be advertised	**Required proof**
ASYM_LINK	packet has been heard
SYM_LINK	ASYM_LINK or SYM_LINK
SYM_NEIGH or MPR_NEIGH	SYM_LINK or SYM_NEIGH
"Node is neighbor"	SYM_NEIGH or MPR_NEIGH

Table 9.1: Required proofs in an ADVSIG message.

- B's link state with respect to A
- the timestamp of the creation time
- the signature (computed by A) of these four fields

The address of the originator node is found in the message header as the `Originator Address` field, and is part of the standard packet. The address of the advertised node and its link state are exchanged through a HELLO message, respectively in the `Neighbor Interface Address` and `Link Code` fields. The timestamp and the signature will be contained in the ADVSIG message coupled to that HELLO. Depending on its use, this atomic information is called either a *Certificate* or a *Proof*.

Hence a node, upon reception of an HELLO and its companion ADVSIG message, extracts from both the information regarding itself (i.e. where "advertised" contain the node's address). When used in this manner, we call the atomic information described above a *Certificate*. The Certificates are stored by the node in a *Certiproof Table*.

Later, when the node sends a HELLO or TC message, it will select the relevant *Proof* from its Certiproof Table and include it in the ADVSIG message coupled to that HELLO/TC message.

Note that we call the same atomic piece of information a Certificate when it is created and supplied to inform about the neighborhood, as fresh reusable topology information; and we call it a Proof when it is reused and supplied to prove a link state. The Certiproof Table of node B contains only Certificates signed by various neighbors of B; in each of these Certificates, the "advertised" field contains the address of B (except in the Certificate Zero, as explained later).

9.3 Required proofs

As mentioned above, if node A wishes to report a link in a HELLO/TC message with a neighbor node B, the required proof must be built using elements of a HELLO message and the accompanying ADVSIG message that were recently sent by node B. The proofs are then stored (as Certificates) in

the Certiproof Table and reused (as Proofs) whenever necessary. The proofs must be sent along packed in a new companion ADVSIG message, with the new HELLO/TC messages they are intended to prove.

Table 9.1 gives a scheme of the required proofs, based on the OLSR specifications [31]. Refer also to Table 1.1 for an explanation of the link codes. The table can be integrated with other link types to define the requested behavior when declaring links of type UNSPEC_LINK or LOST_LINK.

For instance, when A wishes to report (in a HELLO) a SYM_LINK with B, the proof must be a recent HELLO from B reporting an ASYM_LINK or SYM_LINK with A. We remind that link codes (ASYM_LINK, SYM_LINK, SYM_NEIGH, and MPR_NEIGH) are advertised through HELLO messages, and advertising a node as a neighbor is done through TC messages.

For an asymmetric link (ASYM_LINK), the proof is part of the heard packet, because the advertised node does not hear the originator. A previous version of the protocol [130] required no proof for an asymmetric link. This because all critical operations in OLSR concern symmetrical neighbors: MPRs are selected from among the nodes with which there exist a symmetric link, MPR selection is accepted from nodes with which there exist a symmetric link, and TC messages advertise symmetric links only. Asymmetrical links have the sole purpose to (possibly) establish symmetrical links in an immediate future: these symmetric links can (possibly) be established only by an answer from the advertised node. When a malicious node X falsely advertises an ASYM_LINK, the link is maintained as asymmetric and eventually deleted (after expiration of its validity time, which depends on the `Vtime`); except if the advertised node effectively comes in the neighborhood of X, in which case a symmetric link may truly be established.

However, this may lead to an attack where a compromised node X advertises a fake ASYM_LINK with a node Y that X does not hear; Y may be a 2-hop neighbor of X and X may have known about its existence from a HELLO sent by a common neighbor. Node Y may actually hear X's declaration, and therefore it would advertise a SYM_LINK with X instead of an ASYM_LINK as it should be. Hence node Y advertises a false symmetric neighborhood which may be declared in its TCs. If node X advertises a large number of fake ASYM_LINK with several nodes, it is possible that one of these nodes is or moves in the neighborhood of X, making the attack successful.

We present in this thesis a corrected version of the protocol. In this version, a declaration of an asymmetric link requires a proof, called Certificate Zero. A compromised node X can still carry out the aforementioned attack by recycling the Certificate Zero from Y as overheard as a Proof from another node Z, and not as a Certificate from Y as it should be. However, this would cause a delay in X's declaration, and this delay could therefore be detected by using a tighter synchronization and more stringent checks on timestamps.

9.4 The Certiproof Table

When a node B receives from A a HELLO and its accompanying ADVSIG message, it extracts from both any information regarding itself, and stores the tuple

$$\langle originator, advertised, linkstate, timestamp, signature \rangle$$

in its Certiproof Table. This tuple will later be resent by B as a Proof, but the same information was called a Certificate when sent by A. As already explained, *originator* contains the address of A, *advertised* contains the address of B, *linkstate* is B's link state with respect to A, *timestamp* is the time when A generated the HELLO and ADVSIG messages, and *signature* is the signature computed by A on the four fields *originator*, *advertised*, *linkstate*, and *timestamp*.

Note that, in the implementation, it is obviously not necessary to store the *advertised* field in the tuple, as it is a constant. We present nonetheless the protocol in this way to be consistent with the Certificate/Proof format, and to avoid confusion.

The key of the tuple is the *originator* address. Only one tuple for each originator is maintained in the table: when B receives a subsequent HELLO message (with its ADVSIG) from A, it updates the tuple entry with the freshest information, established as such by comparing the *timestamp* fields. In this manner, node B stores in the Certiproof Table only the most recent Certificate about itself, as given by a neighbor.

9.5 The ADVSIG message

The format of this security-enhanced ADVSIG message is shown in Figure 9.2. An ADVSIG message must be generated and sent with every HELLO or TC message, and possibly in the same packet. However, there is a difference between HELLOs and TCs: while both message types always require Proofs, HELLOs can contain Certificates whereas TCs do not. Hence the `Signature of Certificate #`*i* fields exist only in those ADVSIG messages which are coupled to HELLOs.

The `Global Timestamp` is the timestamp of this ADVSIG message and of the HELLO/TC it is coupled with.

The `Global Signature` is computed on the sequence of bits made up of the whole HELLO/TC message (header included) and the associated ADVIG message except, of course, the `Global Signature` field itself. As seen in Section 5, the `Time To Live` and `Hop Count` fields are considered as set to zero, because they change in transit.

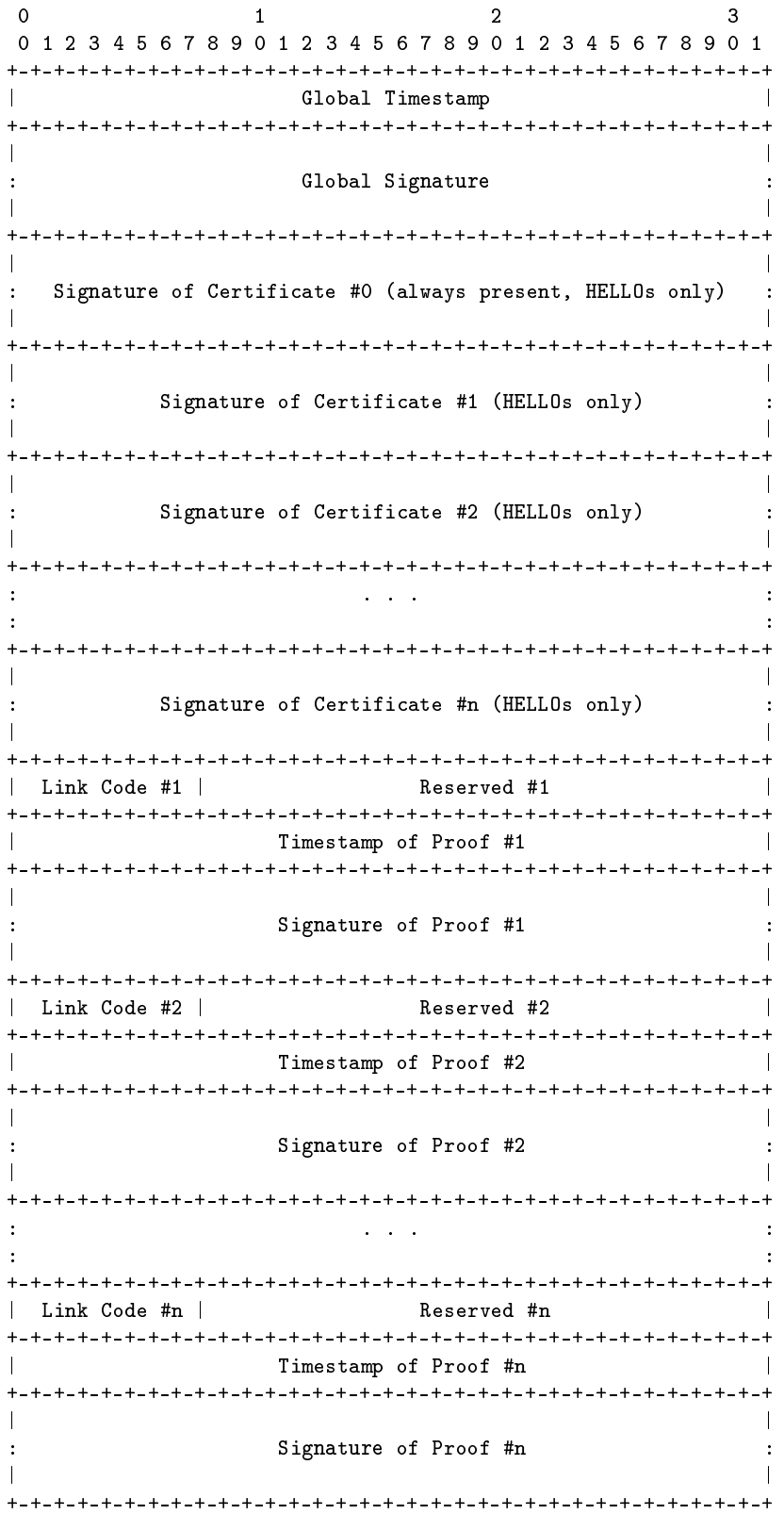

Figure 9.2: ADVSIG message format.

The `Signature of Certificate #`i is present only when the ADVSIG is coupled with a HELLO. This fields contains the signature of the Certificate related respectively to the `Neighbor Interface Address` at position i in the HELLO coupled message.

An exception is the `Signature of Certificate #0` field, which is not related to any advertised neighbor link, but is always included in those ADVSIGs that are coupled to HELLOs.

The subsequent three fields (`Link Code #`i, `Timestamp of Proof #`i, and `Signature of Proof #`i) purport to the Proof related to a neighbor node declared in the HELLO/TC message. More in detail, the Proof is related: to the `Neighbor Interface Address` at position i if the coupled message is a HELLO, or to the `Advertised Neighbor Main Address` at position i if the coupled message is a TC.

The `Reserved #`i field is used to make all fields 32 bit aligned, and may be reserved for future use.

The `Link Code #`i is the link state in the Proof related to neighbor node i. An alternate implementation might omit this field, as it can be extrapolated from Table 9.1; in this case, when verifying a signature of a Proof, a receiver node must test all link codes that apply.

The `Timestamp of Proof #`i and `Signature of Proof #`i are the timestamp and signature of the Proof related to the neighbor node i.

The link status, timestamp, and signature of the Proof were taken respectively from: `Link Code`, `Global Timestamp`, and `Global Signature` of a previous HELLO and its accompanying ADVSIG. These data were then saved in a tuple and stored in the Certiproof Table. If a proof is not required according to Table 9.1, these three fields, as well as the `Reserved #`i field, are not present.

Every `Signature of Certificate` and every `Signature of Proof` is computed on the sequence of bits made up of:

- the relevant `Originator Address` (from the header of the HELLO)
- the relevant `Neighbor Interface Address` (from the HELLO)
- the relevant `Link Code` (from the HELLO)
- the relevant `Global Timestamp` (from the ADVSIG)

The `Signature of Certificate #0` is computed only on the `Originator Address` and `Global Timestamp`, because there is no neighbor to advertise, and hence no link. The purpose of this Certificate Zero is to certify to a neighbor the hearing of an empty HELLO, or a HELLO that does not include that neighbor; in this case the neighbor can issue a declaration of ASYM_LINK

with the sender of the HELLO message. Consequently, the *advertised* and *linkstate* fields are empty in the relevant tuple of the Certiproof Table.

In sum, when A sends an ADVSIG message, every `Signature of Certificate` is signed by A, while every `Signature of Proof` is signed by other nodes (which are, or have recently been, neighbors of A).

9.6 The protocol

In the following example we illustrate the algorithm by scrutinizing the building of a neighborhood. We recall that the notation $A \rightarrow B : \{M, M', T_A(t_0)\}_A$ means "A sends B the message M with the Proof M', timestamped by A at the time t_0, and signed with the private key of A".

1. $A \rightarrow B : \{\{\text{“ ”}, T_A(t_1)\}_A, \varnothing, T_A(t_1)\}_A$
2. $B \rightarrow A : \{\{\text{“}A : ASYM\_LINK\text{”}, T_B(t_2)\}_B, \{\text{“ ”}, T_A(t_1)\}_A, T_B(t_2)\}_B$
3. $A \rightarrow C : \{\{\text{“}B : SYM\_LINK\text{”}, T_A(t_3)\}_A,$
   $\{\text{“}A : ASYM\_LINK\text{”}, T_B(t_2)\}_B, T_A(t_3)\}_A$

In step 1, A sends an empty HELLO, including a Certificate Zero, and no Proof. In step 2, B receives the HELLO from A and declares an ASYM_LINK with B, using the Certificate Zero as proof. In step 3, A declares a SYM_LINK with B; node C is sure that A's statement about its link state with B is correct. To be able to give the Proof in step 3, A stored in its Certiproof Table the tuple $\langle B, A, ASYM\_LINK, T_B(t_2), \{\}_B \rangle$ which was extracted from the data A received from B in step 2: $\{\text{“}A : ASYM\_LINK\text{”}, T_B(t_2)\}_B$.

9.6.1 Implementation of the algorithm

We denote t_0 the current time. The value Δt_g is the time interval after which a `Global Timestamp` expires. The value Δt_p is the maximum acceptable time interval between a Certificate and its Proof, after which the Proof is stale and can no longer be used; this is done in order to thwart replay attacks using old Proofs. (We leave aside the problem of clock skew.) Upon reception of an ADVSIG message, the receiving node must check that the following conditions are satisfied for every k:

- $t_0 - \Delta t_g <$ `Global Timestamp` $< t_0$, i.e. the ADVSIG message is not too old;
- `Global Timestamp` $- \Delta t_p <$ `Timestamp of Proof` $k <$ `Global Timestamp`, i.e. the Proof k is not too old with respect to the HELLO/TC message.

The following subsection outlines the algorithm. The full detailed version of the algorithm is given in the next subsection.

9.6.2 Outline of the algorithm

When a node generates a HELLO or TC message, it must also generate a ADVSIG message, following this protocol:

1. create the HELLO/TC message;
2. write the timestamp;
3. if the message is a HELLO then compute the signature of the Certificate Zero and, for each advertised link, compute the signature of the Certificate and add the relevant required Proof;
4. else if the message is a TC then just add the relevant required Proof;
5. compute the signature;
6. send the HELLO/TC and ADVSIG messages.

When a node receives a control message and its ADVSIG, it must follow this protocol:

1. check the validity of the timestamp;
2. check the validity of the signature;
3. if the message is a HELLO then, for each advertised link, check the validity of the Proof, and extract the Certificate regarding yourself, if any, or the Certificate Zero if there is no Certificate regarding yourself;
4. else if the message is a TC then, for each advertised neighbor, just check the validity of the Proof.

If any of the previous checks fail, the HELLO/TC and ADVSIG message must be dropped.

9.6.3 Detailed algorithm

When a node generates a HELLO or TC message, it must also generate a ADVSIG message, following this protocol:

1. create the HELLO/TC message;
2. write the `Global Timestamp` $\leftarrow t_0$;
3. if the message is a HELLO then
   (a) compute `Signature of Certificate #0` on: `Originator Address` and `Global Timestamp`;
   (b) for each `Neighbor Interface Address` i

i. compute `Signature of Certificate #`i on: `Originator Address`, `Neighbor Interface Address` i, `Link Code`, and `Global Timestamp`;
ii. find the required Proof in your Certiproof Table;
iii. copy the Proof's link state in `Link State #`i, or if this field is empty (i.e. when proving an ASYM_LINK) set the `Link State #`i field to 0;
iv. copy the Proof's timestamp in `Timestamp of Proof #`i;
v. copy the Proof's signature in `Signature of Proof #`i;

4. else if the message is a TC then

(a) for each `Advertised Neighbor Main Address` j
i. find the required Proof in your Certiproof Table;
ii. copy the Proof's link state in `Link State #`i;
iii. copy the Proof's timestamp in `Timestamp of Proof #`j;
iv. copy the Proof's signature in `Signature of Proof #`j;

5. compute the `Global Signature`;

6. send the HELLO/TC and ADVSIG messages.

When a node receives a control message and its ADVSIG, it must follow this protocol:

1. check the validity of `Global Timestamp`;

2. check the validity of `Global Signature`, using the public key of the sender node;

3. if the message is a HELLO then

(a) for each `Neighbor Interface Address` k
i. check the validity of `Timestamp of Proof #`k;
ii. if `Link Code #`k = ASYM_LINK then
A. check the validity of `Signature of Proof #`k computed on: the sender's address and `Timestamp of Proof #`k;
iii. else
A. check that `Link Code #`k correctly proves the `Link Code` of Certificate k (according to Table 9.1);
B. check the validity of `Signature of Proof #`k computed on: the address of k, the sender's address, `Link Code #`k, and `Timestamp of Proof #`k;
iv. if `Neighbor Interface Address` k = your address then
A. extract (from the HELLO) $\lambda \leftarrow$ `Link Code` relevant to `Neighbor Address` k i.e. your link state;

B. store in your Certiproof Table the tuple
⟨ sender's address, your address, λ, `Global Timestamp`, `Signature of Certificate #`k ⟩;

(b) if none of the `Neighbor Interface Address` is your address then

i. store in your Certiproof Table the tuple
⟨ sender's address, $\varnothing$, $\varnothing$, `Global Timestamp`, `Signature of Certificate #0` ⟩;

4. else if the message is a TC then

(a) for each `Advertised Neighbor Main Address` h

i. check the validity of `Timestamp of Proof #`h;

ii. check that `Link Code #`h correctly proves the `Link Code` of Certificate h;

iii. check the validity of `Signature of Proof #`h computed on: the address of h, the sender's address, `Link Code #`h, and `Timestamp of Proof #`h.

If any of the previous checks fail, the node must stop processing the HELLO/TC and the ADVSIG, and must discard them.

9.7 Overhead

We assume the use of DSA to generate the signatures in the ADVSIG message.

The size of an ADVSIG message sent with a HELLO message is:

ADVSIG coupled to HELLO: $352 + 704n$ bits

An ADVSIG message sent with a TC is shorter (because it does not contain Certificates) and has the following size:

ADVSIG coupled to TC: $352 + 384n$ bits

Counting the IP, UDP, and OLSR packet headers, the size of a OLSR packet containing a HELLO or TC message plus its companion ADVSIG message are therefore:

HELLO + ADVSIG (packet): $936 + 744n$ bits
TC + ADVSIG (packet): $832 + 416n$ bits

We assume that each HELLO/TC message and its companion ADVSIG message are sent together in the same OLSR packet, and the packet does not contain other messages.

With the assumptions above and those made in Section 5.4.1 ($n = 9$), the flowrate can be evaluated as follows:

OLSR with ADVSIG: 4731 bit/sec

Figure 9.3 shows the additional message overhead for the ADVSIG architecture.

As can be seen, the message overhead is quite large (even if a large part of this overhead comes from ADVSIGs that are coupled to HELLOs and that therefore do not travel further that one hop in the network). The computation overhead is elevate too, as sending a HELLO message requires $n + 2$ signature computations, while sending a TC requires one; checking the validity of either messages requires $n + 1$ signature verifications. This is unsuitable for low-equipment nodes. For this reason, this protocol may be fit for high-capacity machines in a military network operating in a battlefield, where integrity of topology is of primary importance.

The message overhead can be reduced by using shorter signatures (e.g. Boneh-Lynn-Shacham). By using 64-bit signatures and removing the `Reserved` fields for a more efficient padding, the size of a packet containing a HELLO or a TC plus its ADVSIG would respectively be $680 + 208n$ and $576 + 136n$. The equivalent flowrate would be 1636 bit/sec. Figure 9.4 shows the message overhead for the ADVSIG architecture with shorter signatures.

9.8 Resilience and remaining vulnerabilities

With respect to the vulnerabilities explained in Section 3.2, this architecture protects against an attacker trying the link spoofing attack. This means reaching an important goal. A compromised node can no longer choose the (false) routing information to issue, because this information has to be validated by previous routing information issued beforehand. Hence the network is now robust against one lone attacker. The network is protected even if there are more than one compromised node at the same time, provided that they cannot communicate between them.

If the attacker compromises two or more nodes and is able to have them communicate, it can forge any kind of Certificate or Proof where originator and advertised node are both compromised. Hence any distributed information concerning link state between two compromised nodes may be false. This includes the case in which the attacker is able to create multiple identities of a node (*Sybil attack* [39]), in order to certify the (false) information from a compromised node.

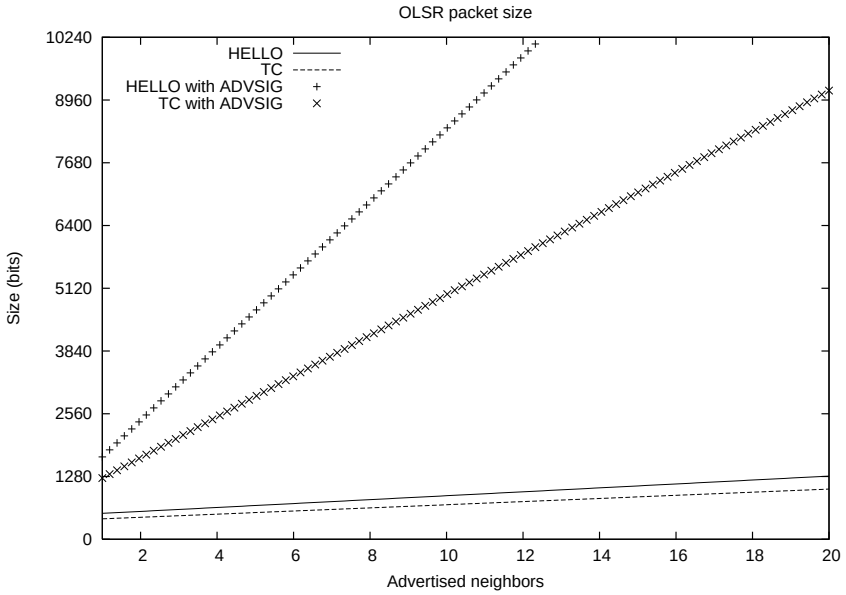

Figure 9.3: Diagram of ADVSIG overhead.

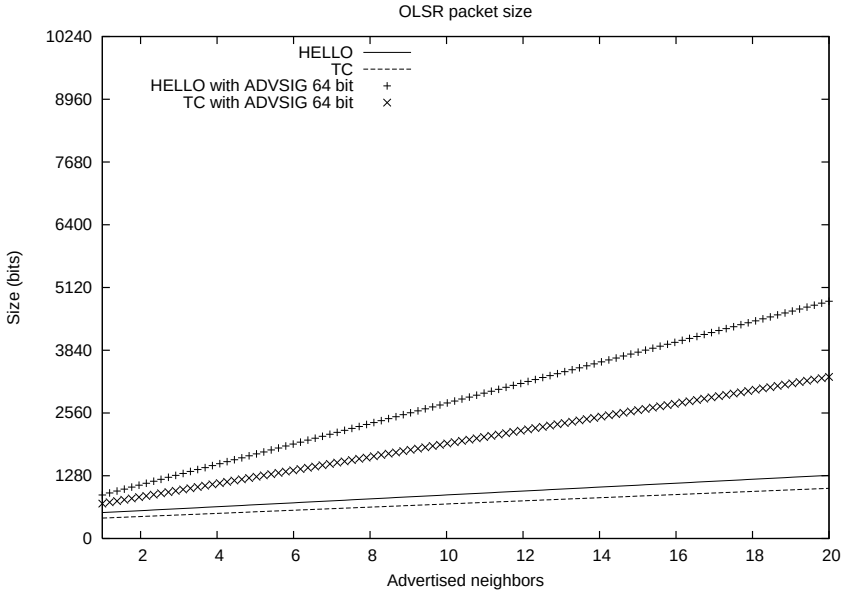

Figure 9.4: Diagram of ADVSIG overhead using 64-bit signatures.

Chapter 10

Using information about node location

Some useful information that can be added in a node's messages, in order to achieve redundancy and strengthen security, is the node's geographical location.

10.1 State of the art

Geographical information is sometimes used for the basic functioning of routing protocols, such as the DREAM (Distance Routing Effect Algorithm for Mobility) protocol [8]. Nodes using DREAM disseminate information about their position by broadcasting a beacon. Then, a sending node uses a probabilistic model to derive a direction in which the destination node is likely to be found. Packets are forwarded in the chosen direction until they reach the intended destination.

GPSR (Greedy Perimeter Stateless Routing) [84] is similar to DREAM, as the sending node forwards the packet towards the node which is closest to the intended destination. The process repeats until the packet reaches the destination. This is the default functioning mode, called Greedy Forwarding. If the default forwarding mode is not possible in a region (due to lack of nodes close enough to the destination) the protocol routes the packet around the perimeter of that region.

LAR (Location-Aided Routing) [88] is another position-based ad hoc routing protocol. The route discovery mechanism uses message flooding, with an optimization to reduce routing overhead: the node initiating the discovery defines geographically a request zone, and only nodes within the request zone forward the route request message.

The protocols mentioned above are not secured, and use geographical information solely for routing. An example of position-based secured routing protocol is SPAAR (Secure Position Aided Ad hoc Routing) [24]. SPAAR

uses public key cryptography to secure hop-by-hop communications, and requires a TTP server for the delivery of certificates. A node A broadcasts its certificate through HELLO messages; one-hop nodes which hear the HELLO respond by sending their certificate, position, and transmission range, encrypted with A's public key. Upon reception, A verifies that the nodes are truly neighbors, and stores their public key, last known position, and transmission range, in its Neighbor Table. Afterwards, A creates a Neighbor Group key pair and distributes the Neighbor Group decryption key to each neighbor. The Neighbor Group encryption key is used later by A to encrypt all its control messages (RREQs, RREPs and RERRs).

The geographical position can be obtained by using Global Positioning System (GPS) devices embedded into the hardware of each node [38]. The GPS is a satellite-based navigation system that makes possible to know the precise position of the device anywhere on Earth or in Earth orbit; the position is extrapolated from the measures of the distances from the device to a minimum of two satellites. The same GPS facility can be used to provide time synchronization [150].

There exist other solutions which do not require every node to be equipped with a GPS device [138] or which do not use GPS at all [153, 46]. These solutions rely on signals or other feedback from other nodes (e.g. the emission power). However, in a network where the presence of malicious nodes is possible, these solutions cannot be considered safe.

10.2 GPS-OLSR

We propose a protocol [131, 132] that enhances security by including and processing the geographical position of the sending node in its control messages. This solution may also be applied to other link state protocols. It is inspired from the work by Hu et al. about packet leashes [67]. Here we assume that the geographical information is obtained by a safe source, like an embedded GPS device.

Several attacks can be thwarted if we possess information about node position, i.e. if every node knows the correct geographical position of any other node in the network. Nodes then compare this geographical data to the received control messages containing topology data (the neighbor and link set). If contradictory information is found, the false control message is detected and discarded.

Besides, the availability of geographical information about nodes in the network opens speculations about possible new features in the standard OLSR, such as improved MPR selection and link breaking forecast. For instance, when two linked nodes are moving in opposite directions (with the distance between the two nodes rapidly increasing), a link break will shortly occur. Therefore, each of the two nodes should not select the other as a

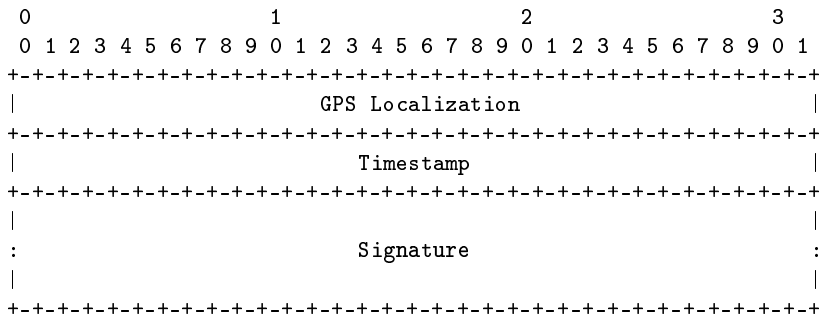

Figure 10.1: SIGLOC message format.

MPR. These issues are not in the scope of this thesis, and are not discussed further.

10.2.1 Specifications

We propose to add a control message called SIGLOC (which stands for SIGnature and LOCalization), based upon the SIGNATURE message presented in Section 5. The SIGLOC message carries an additional `GPS Localization` field which contains the current geographical position of the sending node, as obtained from the GPS facility, and is included in the signature computation. This field is 32 bits long, that is enough to define the position over an area of more than 4200 square km with a precision of 1 m using Cartesian coordinates; a more efficient representation can also be used.

The format of a SIGLOC message is given in Figure 10.1. All other fields, as well as the mechanisms of signature computation and verification, are the same as for the SIGNATURE message. Similarly, a SIGLOC message is generated and sent along with each HELLO or TC message.

A node informs the other nodes about its current geographical position via this SIGLOC message. The receiving node verifies the correctness of the timestamp and signature as previously specified, and extracts the timestamp and the information relative to the position of the originator node. This data is stored as a tuple

$$\langle address, position, timestamp \rangle$$

in a *Position Table* maintained by each node. The Position Table has the purpose to memorize the most recent position of every other node in the network. Note that geographical information is propagated in the network via SIGLOCs coupled to TCs, as HELLO messages are not spread around further that one hop. How this geographical information is utilized to improve security will be discussed in Section 10.2.2.

The advantage in knowing the geographical position of nodes is that we can speculate whether communication of a message from a sender node S is likely to be heard or not by a receiver node R. Let p_S and p_R be the current position of the sender and the receiver and T_R the current time according to the receiver's clock. Also let Δt be the discrepancy in the clocks' synchronization, Δd the maximum absolute error in position information, v_{max} the maximum velocity of any node, and r_{max} the maximum transmission range. Based on the timestamp T_S of the sender's message, we can compute a lower bound on the distance d_{SR} between the sender and the receiver. In fact it must be

$$r_{max} \geq d_{SR} \geq \|p_R - p_S\| - (T_R - T_S + \Delta t) \cdot 2v_{max} - \Delta d \qquad (10.1)$$

as shown in Figure 10.2, where the radius r of the circle is the quantity on the right of the formula: $r = \|p_R - p_S\| - (T_R - T_S + \Delta t) \cdot 2v_{max} - \Delta d$. If the (10.1) is not valid, this means that the receiver node is too far from the sender node to be able to hear its transmission; therefore such a transmission is highly suspicious and might be a fake.

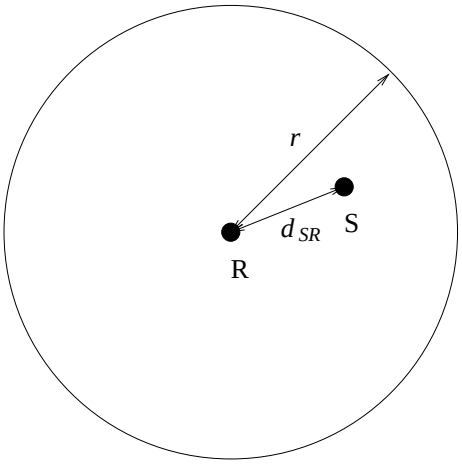

Figure 10.2: Lower bound on the distance between R and S.

An important remark: We denote as sender S the last-hop node that emitted the message received by R; this means that receiver R is 1-hop far away from sender S, i.e. R and S are neighbors. In the case of messages (e.g. TCs) that are being relayed, the sender node S is not the same as the originator node, which is the node that created the message. The address of the originator is contained in the `Originator Address` field in the HELLO header, and does not change while the packet is relayed around the network. The sender address S is the source address from the IP header of the packet, and is changed, each hop, to the address of the node which is retransmitting the message.

Note that the standard OLSR already defines some checks to be performed at message reception; if the sender S of a TC/MID/HNA message is not a symmetric neighbor of receiver R, the latter must drop the message.

10.2.2 Resilience

We believe this protocol to be robust against two of the most severe attacks: link spoofing and wormhole.

It may be argued that a compromised node X can forge the GPS information contained in the SIGLOC message. The compromised node can choose any value from scratch; in case the GPS device is tamperproof and supplies geographical data in an encoded format, node X can even record other node's geographical data, or its old own, and reuse it later. In this case, X could pretend being over time in very different parts of the network, and advertise links with nodes which are in its current part of the network, in order to perform a link spoofing attack.

However, this does not work because node X (as any other node) is always bounded by its maximum velocity v_{max}. A node that has in its location table the tuple $\langle X, p_X, T_X \rangle$, and receives a SIGLOC message from X carrying a geographical data p'_X and a timestamp T'_X, must check if the following condition holds true:

$$\|p'_X - p_X\| \geq (T'_X - T_X) \cdot v_{max} \tag{10.2}$$

If the (10.2) is not valid, this means that node X pretends to be in a location it could not reach according to its maximum velocity; therefore either p_X or p'_X are likely to be false.

Protection against link spoofing

For any communication between a sender and a receiver, the Formula 10.1 must hold valid and this obviously also applies to link state. We can therefore detect the case in which a misbehaving node X falsely advertises a link (in a HELLO message) with the non-neighbor node N, or declares N as a neighbor (in a TC message). In the case of such a false declaration, the (10.1) is in fact not valid with respect to the distance d_{XN} as evaluated by the receiver A of the message (Figure 10.3).

Protection against wormhole attacks

When a message is being maliciously tunneled between legitimate nodes A and B, the Formula 10.1 is not valid with respect to the distance d_{AB} as measured by A (Figure 10.4).

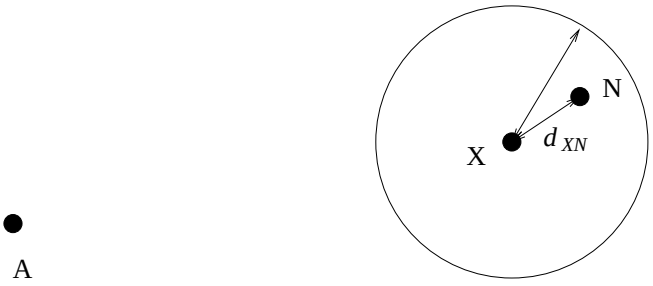

Figure 10.3: Test of likelihood for declared links.

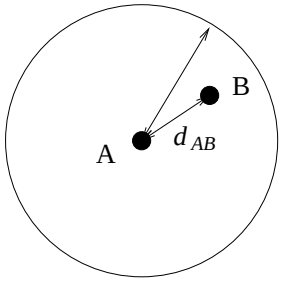

Figure 10.4: Test of likelihood for a direct link (against a wormhole).

10.2.3 The protocol

A node generates a SIGLOC message together with any generated HELLO or TC message, with the same specifications of the SIGNATURE message referred in Section 5. The following step is added to the protocol:

- the node writes in the `GPS Localization` field the geographical data concerning its actual position, as obtained from its own GPS device.

When node B receives a SIGLOC and its HELLO/TC from node A, it handles them in the same way it does with a SIGNATURE message, performing the same tests (match of the SIGLOC with the companion HELLO/TC, timestamp validity check, and signature verification). Note that A is the last hop to B, and that we call O the originator of the control message relayed by A. If the control message is a HELLO, then O is surely A because we know HELLOs are not relayed; if the control message is a TC, then O may or may not be A (depending whether A is an MPR or not). The following steps are added to the protocol:

- node B checks that Formula 10.2 is valid with respect to A;
- for each neighbor N declared in the HELLO/TC by the originator node O, node B checks that Formula 10.1 is valid with respect to d_{ON};

- node B updates the entry concerning O in its Position Table, storing the tuple $\langle O, \texttt{GPS Localization}, \texttt{Timestamp}\rangle$ with the actual values from the SIGLOC.

If any of the previous checks fail, node B must stop processing the HELLO/TC and the SIGLOC, and must discard them. Note that this algorithm assumes that B has entries for A, O, and N, in its Position Table; this may not be the case at network initialization. Immediately after network bootstrap, during the first formation of the network, if B lacks the entry needed for the test it should therefore bypass the relevant step.

10.3 Using a directional antenna to obtain extended accuracy

Using a directional antenna instead of an omni introduces additional security. With that, node R can know from which direction the signal is coming; by basing on p_R and p_S and using simple plane geometry, R can check roughly the correctness of the value p_S. Let $[\theta, \Delta\theta]$ be the sector of R's antenna on which the signal is received, and denote with (d_{SR}, θ_S) the coordinates of S in the polar coordinate system with origin in R. In addition to (10.1), the following condition must also be true:

$$\theta \leq \theta_S \leq \theta + \Delta\theta \tag{10.3}$$

as shown in Figure 10.5. Formula 10.3 is useful even if the maximum transmission range r_{max} is not known with precision.

10.4 Numerical evaluation

We analyze the consequences of the (10.1). With figures such as $v = 60$ km/h, $T_R - T_S + \Delta t = 100$ msec, and $\Delta d = 1$ meter, we obtain $\|p_R - p_S\| \leq r_{max} + 4.333$ meter. When r_{max} is not too small (e.g. $r_{max} > 50$ meter), the received packet is necessarily sent from a nearby node within the coverage of the recipient. Therefore, wormhole attacks tunneling such a packet would be difficult to accomplish because the real packet is likely to be heard by the recipient; on the other hand, such an attack would be not very efficient, since the node whose packet is relayed is, most likely, a few hops away. When r_{max} is small (e.g. 20 meter $< r_{max} <$ 50 meter) the information given by a directional antenna can be useful, since the sector in which the signal is expected has a limited size.

10.5 Overhead

We assume the use of DSA to generate the signature in the SIGLOC message.

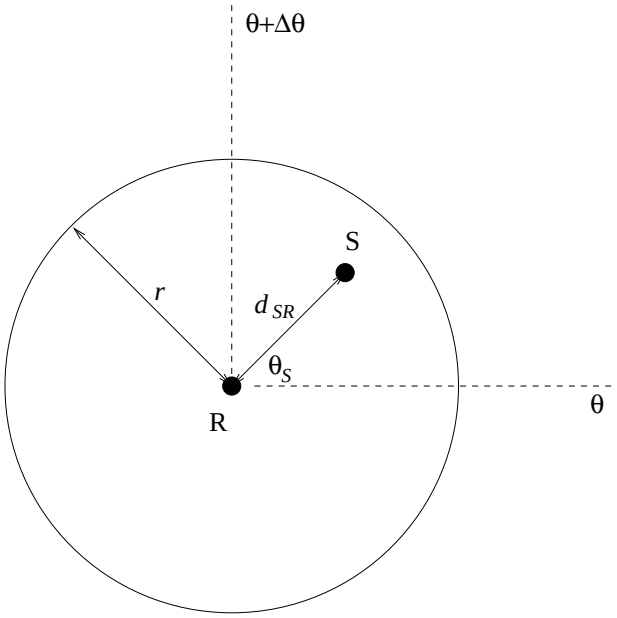

Figure 10.5: Direction check on the distance between R and S.

The size of a SIGLOC message is:

SIGLOC: 384 bits

Counting the IP, UDP, and OLSR packet and message headers, the size of a OLSR packet containing a HELLO/TC message and its companion SIGLOC are:

HELLO + SIGLOC (packet): $968 + 40n$ bits
TC + SIGLOC (packet): $864 + 32n$ bits

We assume that each HELLO/TC message and its companion SIGLOC message are sent together in the same OLSR packet, and the packet does not contain other messages.

With the assumptions above and those made in Section 5.4.1, and considering a neighborhood of 9 nodes, the flowrate can be evaluated as follows:

OLSR with SIGLOC: 894 bit/sec

Figure 10.6 shows the additional message overhead for the SIGLOC architecture.

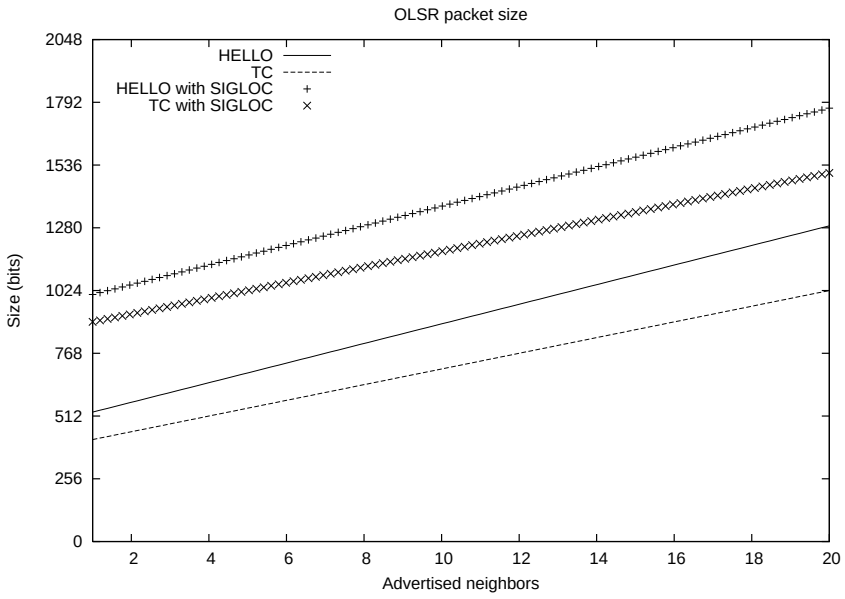

Figure 10.6: Diagram of SIGLOC overhead.

Chapter 11

Detecting bad behaviors

While it is important to strengthen security in order to prevent network intrusions, and therefore misbehaviors, it is also useful to use audit tools to detect these possible misbehaviors in the network. As remarked by Schneier [140], a prevention-only strategy works only if the prevention mechanism is perfect. In this case, we accept the possibility that, despite the security measures, an intrusion can be successfully carried out; what we want now is to identify the intruder as soon as it starts perturbing the network, and neutralize it. The misbehaving node can be a former legitimate node that has been compromised, or an intruder node that managed to join the network. Note that node misbehavior may also be not due to malice, e.g. in the case of malfunctioning or battery exhaustion. Nevertheless also these misbehaviors should be detected and stopped, since they can damage the correct functioning of the network.

Once a node detects a misbehaving node, it alerts the network. Upon reception of the alert, the other legitimate nodes should take a corrective action to exclude the misbehaving node to participate further in the network.

The techniques mentioned can be used with or without an infrastructure to sign messages, i.e. both on secured or unsecured routing protocols. They do not replace message signatures and other security measures; they work in parallel to them. Of course, the danger in not using signatures is that an attacker could revert the technique against the network, falsely accusing well-behaving nodes and/or issuing alerts with a spoofed originator.

11.1 State of the art

Different ways exist for countering attacks such as the blackhole (in which, we recall, a node causes a Denial of Service by failing forwarding packets in accordance to the protocol) or similar. An example is to overhear transmissions to detect incorrect forwarding behavior, as in the Watchdog/Pathrater [104] or in CONFIDANT [20]. Bloodhound [103] is a modified version of the

Watchdog, designed in order to patch a security flaw in SRP which makes SRP vulnerable to the invisible node attack. Another issue is the WATCHERS [19, 70] approach based on the principle of conservation of flow. Other ways use detection through acknowledgements [122] or probe packets [26].

11.1.1 Watchdog/Pathrater

The Watchdog/Pathrater system is designed as an extension to DSR. It is composed of a module called Watchdog that identifies misbehaving nodes, and a module called Pathrater that computes a route avoiding these nodes. These modules are run by each node in the network.

The Watchdog listen promiscuously to the next node's transmission, checking that the node correctly forwards the packet it has received. The Watchdog can also detect if the node has tampered with the payload. This is done by comparing the listened packet to a buffer of recently sent packets. The Pathrater processes the information obtained from the Watchdog to rate the reliability of every other node it knows in the network, and to calculate a path metric obtained by averaging the node ratings in the path. The packets are then routed through the path with the highest metric.

This system cannot be reverted against the network because such behavior would be easily detected. In a path $A - X - B - C - D$, node X (misbehaving) could falsely report that node B is not forwarding packets. However, the acknowledgement of a message from A to D travels correctly from D to A (node X cannot drop neither packets nor their acknowledgement because A and B would detect this misbehavior), and then A becomes aware that B is not misbehaving because it is included in the path.

We consider a path $A - B - C$. The weaknesses of this system is that the Watchdog running in node A may fail in identifying a misbehaving node in some condition.

- A packet collision may occur in A when A is listening to B. In this case A cannot know whether the collision was due to B forwarding the packet (well-behaving), or to another node transmitting while B did not forward the packet (misbehaving);
- When A is listening to B forwarding to C, it detects that B sends correctly the packet. However, node A cannot detect whether C received it, or a collision occurred in C and B did not re-send the packet (misbehaving);
- Node B may tweak its transmission power (misbehaving) so that A detect that B is forwarding a packet to C, but C does not receive it;
- Nodes B and C (both misbehaving) may collude to mount an attack. Node B forwards correctly a packet to C, but does not report C dropping the packet;

- Node B may drop packets (misbehaving) at a lower rate than the minimum misbehavior threshold of A's Watchdog.

11.1.2 CONFIDANT

CONFIDANT (Cooperation Of Nodes: Fairness In Dynamic Ad hoc NeTworks) is a protocol similar to the Watchdog/Pathrater. It is developed with analogy to an ecological system, where natural selection ensures the survival of those elements of population who bears a grudge against the elements which does not behave altruistically. Hence, CONFIDANT too is based on cooperation between well-behaving nodes, and isolation of misbehaving nodes.

The protocol is composed of four modules which are run in each node: the Monitor, to listen to neighbors' transmissions and identify misbehaviors; the Reputation System, to rate the nodes; the Path Manager, to create and delete paths according to a security metric derived from the nodes' ratings; and the Trust Manager, to manage the trust level of the nodes and issue alarms.

11.1.3 WATCHERS

The *principle of conservation of flow* in a network states: "All data bits sent to a node, and not destined to that node, must exit the node", or "An input must either be absorbed or sent on as an output". WATCHERS (Watching for Anomalies in Transit Conservation: a Heuristic for Ensuring Router Security) is a distributed protocol based on the principle of conservation of flow. Each participating node checks that incoming packets have been correctly routed, and counts the data bits passing through neighbor nodes. The results are periodically reported to other participating nodes, in order to allow each node to check if its neighbors have respected the principle. If a node finds that one of its neighbors is misrouting packets or violating the principle, it stops sending packets to the misbehaving node.

A drawback of this method is that proving that the proper number of packets was forwarded by a node, does not prove that the proper packets were sent, as it cannot be proven that packets have not been tampered with.

11.2 A trust system for OLSR

As said previously, a node can be able to notice if its neighbor is not forwarding packets; in this case, the noticing node may take appropriate action and propagate the alert to the other nodes.

However, a critical problem when exchanging trust evaluation between nodes is how to distinguish false alarms from good ones. Compromised nodes may issue false alarms regarding legitimate nodes, in order to exclude

them from the network and therefore cause a Denial of Service. In the CONFIDANT protocol, the problem is lessened by timeout and subsequent recovery of nodes that have behaved well for a specific period of time.

The problem lies in the difficulty to evaluate node A's statement about B. If A states that B is misbehaving, is A a good node that reports B's misbehavior, or is B good and A is compromised? Hence the situation is symmetric. In fact, once a node has fallen in enemy's hands, as long as its messages strictly follow the protocol and the node uses its private key to sign its messages, these messages appear to be perfectly valid (and in fact they are) and are accepted by the rest of the network. We nevertheless present here our approach, which could partly solve this dilemma, and which consists in a balanced accusation system and link removal.

In our opinion, there is a criterion that allows us to distinguish between compromised and non-compromised nodes: as cracking a PKI costs highly in terms of efforts and time, we may suppose that compromised nodes are outnumbered by good nodes. We can then use this advantage by requiring that alerts must be confirmed by more than one node. There is safety in number. With this approach in mind, we outline a protocol for misbehavior detection, which imitates the Pathrater and which uses an evaluation and rating system of accusations.

11.2.1 Specifications

A global *Trust Table* lists each node in the network, with a numerical value associated to each node and representing its level of trust. This value means how much this node should be trusted to be well-behaving. Every node maintains a local copy of the Trust Table.

When a node detects a neighbor misbehaving, it immediately alerts the network by broadcasting an accusation message containing:

- the node's own address
- the address of the accused misbehaving neighbor
- a code stating the type of misbehavior
- a timestamp
- the signature

The possible format of an accusation message is specified in Figure 11.1.

Every node also keeps an *Accusation Table* storing all past heard accusations broadcast on the network. Upon reception of an accusation, a node handles and evaluates it independently. This avoids the need of maintaining a centralized entity (which could be a weakness) evaluating the trust. If broadcast is done properly, each node should have the same copy of the Accusation Table.

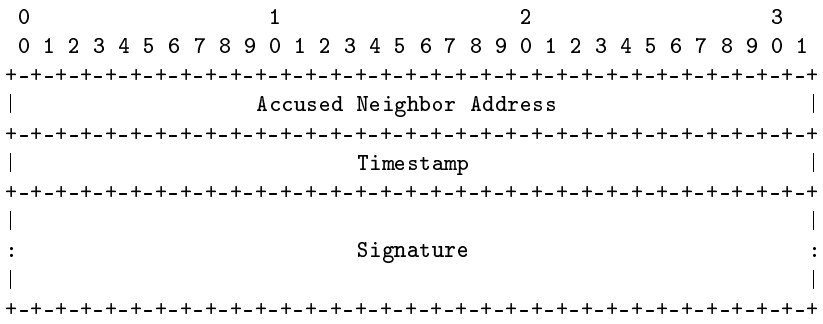

Figure 11.1: Accusation message format.

Optionally, the accusation message could contain also a valid Proof, as described in Section 9, consisting of a HELLO message reporting a symmetric link with the neighbor node. This of course can be done only if the accusation system is implemented over the ADVSIG infrastructure. The advantage of the Proof is that it certifies the accuser node being in the neighborhood of the accused node, and therefore limits the damage a compromised node could do by maliciously accusing non-neighbor nodes (possibly, every other node in the network).

11.2.2 Punishment and reward

A node has a *trust rating* of τ_s at the boot-up. This trust rating changes according to the behavior and reported accusations against the node.

If a node has at least n_i accusations against him (all accusations coming from n_i different accusers) within a time interval t_i, its rating is decreased by α_i and its accusers are rewarded by increasing their rating by ρ_i. However, if within t_i the network fails to find n_i accusations against the node, these accusations are dropped; in this case the node's rating is increased by ρ_i' and the m nodes which accused it ($m < n_i$) are punished by decreasing their rating by $\frac{\alpha_i'}{m}$. With this, we implement a reward and punishment system which remunerates nodes which behave correctly and sincerely report misbehaviors, and punishes false accusers in the opposite way. Each set i of values can be decided depending on the type of misbehavior detected: see Table 11.1 for an example.

When the trust level of a node drops negative, the node is considered as bad (i.e. compromised or malfunctioning) and specific actions should be taken against it. The list of possible actions to take is discussed in Section 11.2.3.

An accusation from a node is automatically discarded if the node has already sent an accusation (as reported in the Accusation Table) within a time $\tau_i \geq t_i$.

Index	Misbehavior
$i = 0$	Failure in reporting a misbehavior
$i = 1$	Failure in forwarding
$i = 2$	Malformed control message
$i = 3$	Stale timestamp
$i = 4$	Invalid signature
$i = 5$	Identity spoofing
$i = 6$	DoS (message bombing) attempted

Table 11.1: Misbehaviors, in order of increasing severity.

As packet collisions and transmission errors happen even on a network where no node is compromised, a well-behaving node might be accused and eventually have its trust rate dropping below zero after a finite time. To avoid this, the rating of all nodes is raised by a bonus β_e every time interval t_e. The values β_e and t_e are computed depending on the probability p_e of a packet collision or transmission error. These values should be set once at network boot-up and not changed anymore, as otherwise this could make the network automatically adjusting as more and more nodes become compromised.

The value of the n_i variable must be chosen as proportional to the density of the network, and depending on the balance we want between different kinds of protection. An attacker could compromise a number of nodes, use them to surround a good node and accuse it until the good node is considered bad, then pass to another good node and so on. A high value for n_i means that a bigger number of accusers is needed against a presumed misbehaving node. Therefore, the network is better protected against the aforementioned attack, as the attacker needs to compromise a bigger number of nodes. However, we must take into account the fact that the network might fail in finding n_i witnesses at the same time t_i for the same event, and therefore that more misbehaviors are unreported. On the other hand, setting a low value for n_i causes more misbehaviors to be detected, but makes the aforementioned attack easier to accomplish.

11.2.3 Detection of a misbehaving node: countermeasures

We have now spotted a node that, with high probability, is misbehaving due to compromission or malfunctioning. However we are not sure which of the two possibilities is correct. A malfunctioning node may fail to route packets, but its other functionalities like message emission may or may not be affected as well. On the other hand, a compromised node is likely to behave as much maliciously as it can, and should therefore be removed from the network definitely and as soon as possible.

As a safety rule, we may choose to exclude the misbehaving node, and

deny routing packets through it. This exclusion is operated by requiring that all routes containing the guilty node are removed from all routing tables. This exclusion may be definitive or may be revoked after a complete recovery over the failing node, e.g. after human intervention.

11.2.4 Variations on the theme of trust evaluation

Another strategy is evaluating trust locally instead of globally; this involves performing link removal instead of node removal. This strategy states that in the case of a successful accusation, the link between the accuser and the accused node must be removed. This is done by the accusing node, by removing the misbehaving node from its routing tables and not accepting messages anymore from the misbehaving node. In this case, there is no need to broadcast accusations; a different Trust Table is maintained independently by each node, and the boot-up trust rate τ_s is set to a smaller value. This strategy is probably safer, as it better maintains network availability. As we do not know whether of the two nodes is misbehaving, we cut the link between them. Should a node misbehave for more than a short time, it isolates itself from the rest of the network.

Another criterion, unsuitable for OLSR but which could be applied to a source routing protocol, is to use the trust rating as a routing metric. A route rating is calculated as the sum (or the average, or another appropriate mathematical operation) of the trust ratings of all nodes included in the path, and the route with the highest rating is chosen. This is similar to the SAR protocol [163] which organizes nodes in a trust hierarchy and incorporates security ranks of nodes into routing metrics; the difference is that trust ratings in SAR are not dynamic.

11.2.5 Precise checks on flow conservation

An additional measure for misbehavior check, which requires all traffic to be authenticated, applies the principle of conservation of flow enunciated in Section 11.1.3 to perform precise checks on flows. Network flow conservation checks are the basis for a set of algorithms that essentially count packets received and sent by a node to each of its neighbors. These counts verify if the node is exhibiting proper routing behavior. The total number of packets which come into a node to be relayed should be equal to the total number of relayed packets coming out from the same node.

When applied to two neighbor nodes A and B, the principle of conservation of flow states that the number of packets sent by A to B must be equal to the number of packets received by B from A.

Now, when wanting to verify the behavior of a third node C being both A's and B's neighbor, the packet counts should be consistent considering each pair involving C and its neighbors, i.e. $C-A$ and $C-B$. Additionally,

nodes A and B should compute statistics about packets in transit through C, where a transit packet of C is a packet that is neither destinated to nor originating from C. On a node C, the principle of conservation of flow hence translates into "The sum of the number of transit packets sent to C by all other nodes must be equal to the sum of the number of transit packets sent by C to all other nodes".

Therefore, a more reliable misbehavior detection is obtained through precise flow conservation checks. Each pair of neighbors (X, Y) records the following information about the packets from X to Y:

- The number of packets that are transit packets for both X and for Y (i.e. neither X nor Y are source or destination): $r_{X \to Y}$.
- The number of packets with source X that are transit packets for Y: $s_{X \to Y}$
- The number of packets with destination Y that are transit packets for X: $d_{X \to Y}$.
- The number of packets that are misrouted by X to Y (i.e. packets forwarded by X to Y although Y is not closer to the destination than X): $m_{X \to Y}$
- The total number of all packets without regards for the source or destination: $a_{X \to Y}$

Each quantity can be seen either from the point of view of node X or node Y. For instance, $r_{X \to Y}$ is denoted $r_{X \to Y}[X]$ for X's perspective, and $r_{X \to Y}[Y]$ for Y's perspective. Similarly, $m_{X \to Y}[X]$ is the number of packets misrouted by X to Y from X's perspective, which should normally be 0.

The complete relation for flow conservation in a node Z is therefore specified as: output − input = produced packets − consumed packets. This quantity is not necessarily zero. Therefore we have that the number of packets sent by Z to neighbors, minus the number of packets sent by neighbors to Z, is equal to the number of packets sent by Z originating from Z, minus the number of packets sent to Z destinated to Z, minus the number of packets sent to Z that Z judged misrouted.

This can be translated into Equation 11.1 for a node Z and the set of its neighbors N_i:

$$(\sum_i a_{Z \to N_i}[N_i]) - (\sum_i a_{N_i \to Z}[N_i]) =$$
$$(\sum_i s_{Z \to N_i}[N_i]) - (\sum_i d_{N_i \to Z}[N_i]) - (\sum_i m_{N_i \to Z}[N_i]) \quad (11.1)$$

The weakness of this technique is that it ensures the proper number of packets is exchanged, but it does not make any assumption about the content

of the packets. An expensive possible solution would be to keep a digest list or Bloom filter [14], instead of a counter, about the packets received and sent.

11.3 A last word about enforcing security

An important point that should not be forgotten is that security often adds redundancy to an architecture, and the burden it gives may be exploited by an adversary to cause damage to the system. For instance, where a secure protocol, which implements message signatures, is deployed, an attacker may send a large number of malformed signatures over the network, in order to keep the nodes busy verifying the signatures and therefore to perform a Denial of Service. This problem would not exist in the non-secure version of the protocol, where the nodes would simply discard the malformed messages sent by the attacker. The following scheme, adapted from Garfinkel [51], shows how each new security countermeasure can be exploited for a new attack, by illustrating at each step a different profile of the attacker and the defender.

Defender: An ad hoc network running the standard non-secure OLSR.

Attacker: An intruder node sending false routing messages to perturb nodes' routing tables.

Defender: An ad hoc network running OLSR with signatures (Chapter 5).

Attacker: A compromised node sending routing messages with a spoofed originator address to perturb nodes' routing tables.

Defender: An ad hoc network running OLSR with the SIGLOC infrastructure (Chapter 10).

Attacker: A compromised node that stops relaying messages to perturb network connectivity.

Defender: An ad hoc network running OLSR with the SIGLOC infrastructure, and a WATCHERS-based system for detection of flow conservation (Section 11.1.3).

Attacker: A compromised node sending false accusations against its neighbors.

Defender: An ad hoc network running OLSR with the SIGLOC infrastructure, a WATCHERS-based system, and an accusation system for detection of misbehaviors (Section 11.2).

This simply means that the security level must be tuned according to the previsible attacks and to the desired level of protection: a military network would obviously need a greater defence infrastructure than that of a public network. Furthermore, securing a network (or, in a general way, a system) is a dynamic process. This process must make use of several and different tools, in order to be ready to counteract different types of assaults.

It should also be noted, as shown by the elusive and elegant jellyfish attack (Section 3.1.1), that misbehaviors may be very difficult to detect. We give here an example of a very theoretical attack against OLSR that, while strictly respecting the protocol, has the effect of muting a node completely. The OLSR protocol adds an amount of jitter to the interval at which control messages are generated. This is done in order to avoid emitting messages at the same time, and hence provoking packet collisions. We may safely assume that, in any implementation of OLSR, the jitter is randomly generated using a PRNG (Pseudo Random Number Generator)[1], which does not give real-random numbers. If the implementation of the PRNG is not carefully chosen, an attacker could replicate its results and command a compromised node to use the same jitter as that which a neighbor node is using. As a consequence, the compromised node could synchronize its transmissions with those of its neighbor, causing message collisions and impeding the neighbor from communicating.

This behavior, while being fully protocol-compliant, can severely degrade the network functioning. Nonetheless, it is an impractical and nonrealistic attack. Denials of Service can be carried on the physical layer as well (e.g. by radio interferences), where they are much easier to carry out.

[1]By their very definition, a random number is unpredictable, whereas the output of a computer algorithm is predictable. Therefore, a computer algorithm cannot generate random results. A PRNG program uses internal sources of pseudo randomness, such as environmental noise from device drivers, and generates only pseudo random numbers. For many applications that do not need high levels of security, pseudo random numbers are suitable; however, critical cryptographic applications should use external sources of real randomness, such as radioactive decay, cosmic rays, or thermal noise in electric circuits [51].

Chapter 12

Conclusion

In this thesis we have provided an overview of the security problems in wireless networks, focusing on the routing protocols in ad hoc networks, and contributed with some solutions to make OLSR more secure. Wireless ad hoc networks are an emerging technology, and the literature covering the aspects of the security of the routing layer is relatively new, the first papers on this subject having been published a few years ago.

The thesis provides a classification of the attacks against OLSR, which is a topic that has never been studied at this level of detail. We have also proposed several solutions for OLSR, these solutions including at first the addition of a digest or a digital signature to control traffic; this is the canonical protection against intrusions in the routing protocol. More elaborate techniques presented in this thesis focus on the validation of link state information, to avoid compromised nodes issuing false information. This is an advanced level of protection, and assumes that an adversary is able to generate correct signatures for control traffic originating from some nodes. These advanced techniques use additional knowledge, such as previous link state declarations or geographical data concerning the position of nodes, to validate the topology information spreaded in the network by the nodes. The increased security is at the expense of a greater message overhead, as exchanged control messages have of course a larger size and implicate further computations done by both the originating and the receiving node. This may be unsuitable for a network composed of nodes that do not have a sufficient computational power, for a QoS-aimed network that must guarantee high performances concerning the data rate, or for a network that simply does not need such an improved security. On the other hand, these techniques can be associated in order to provide an higher security level.

These systems are aimed at the protection of network topology information. Ad hoc networks are the most adaptable and serviceable type of wireless network; for this reason, they are widely used by the military. In this instance, topology information is of great value, and the network should

be protected against intrusions which would have severe consequences.

In addition to the prevention techniques mentioned above, we have also sketched a method for misbehavior detection and elimination. This method aims at detecting those nodes that, by non-respect of the protocol rules, perturb the network functioning. Once that these misbehaving nodes are detected, an alert is broadcast to inform the rest of the network. The other nodes subsequently issue a joint reaction to purge the network of the offending nodes, e.g. by removing them from the routing tables. Of course, this detection system can (and should) be combined with some of the aforementioned prevention techniques.

12.1 Foresights

During our doctoral researches we have found some systems, of different requirements and specifications, to secure OLSR. Other systems may be found by adapting various security techniques and established standards, such as IPsec, always bearing in mind that ad hoc networks have their own characteristics and limitations. These miscellaneous security techniques may also come from other link state protocols, or even reactive routing protocols, with the necessary modifications to conform to OLSR.

Indeed, we have provided just an outline of the signature algorithms utilized in our security systems. The study of better cryptographic algorithms (from the point of view of a smaller signature size, reduced computation complexity, and greater speed) would increase the suitability of the proposed OLSR security architectures to the reality of an ad hoc protocol.

Appendix A

Résumé détaillé de la thèse

Cette thèse traite le problème de la sécurité dans le protocole OLSR pour les réseaux ad hoc. Elle étudie les attaques possibles et propose différentes infrastructures pour la sécurisation d'OLSR.

A.1 Introduction aux réseaux sans fil

Dans les réseaux sans fil les ordinateurs communiquent soit à travers les ondes radio, soit au moyen des rayons infrarouges. Les ondes radio sont le support le plus utilisé: les fréquences disponibles se trouvent dans la bande des micro-ondes, autour de 2.4 GHz (bande ISM) et 5 GHz (bande U-NII). Selon la fréquence, la puissance, le débit de transmission et l'antenne utilisée, la portée d'émission d'une machine peut varier entre 10 et 100 mètres, avec un maximum d'environ 10 Km.

Les standards pour le support hertzien, au jour d'aujourd'hui, sont le IEEE 802.11 avec ses branches principales 802.11a (Wi-Fi5) et 802.11b (Wi-Fi), HiperLAN de l'ETSI, et Bluetooth.

Un réseau sans fil peut fonctionner en différentes modalités: en mode infrastructure ou BSS, où les machines (*nœuds*) sont en connexion à travers un point d'accés; en mode point à point, ad hoc ou IBSS, où les machines communiquent directement entre elles; ou comme un réseau ad hoc ou MANET, où toute machine peut communiquer avec n'importe quelle autre, grâce au fait que les paquets sont relayés par les machines jusqu'à ce qu'il joignent leur destination.

Un réseau sans fil est beaucoup plus souple que son homologue filaire, dans la mesure où les nœuds ne sont pas connectés par de câbles et peuvent être totalement mobiles. Pourtant, un réseau sans fil possède des faiblesses en ce qui concerne la connectivité des nœuds et la sécurité des données.

A.1.1 Les protocoles de routage pour les réseaux ad hoc

Dans un réseau ad hoc, pour permettre la connectivité entre l'émetteur et le récepteur d'un paquet, un protocole de routage doit nécessairement tourner dans tout nœud du réseau. Les protocoles de routage peuvent être classés dans trois catégories:

- dans un protocole *réactif* ou *à la demande*, la demande d'une route pour une destination declenche la recherche d'une route. Parmi les examples de protocoles réactifs on peut citer DSR, AODV et DSDV;

- à l'inverse, un protocole *proactif* ou *périodique* est caractérisé par l'échange périodique des tables topologiques, ainsi les routes sont disponibles d'une façon immédiate. Rentrent dans cette catégorie les protocoles tels que OLSR, OSPF, FSR, TBRPF, ADV, STAR, LANMAR, WRP et WIRP;

- un protocole *hybride*, enfin, utilise les deux systèmes pour le routage. ZRP et CBRP sont des exemples d'un tel type de protocole.

A.1.2 Le protocole OLSR

OLSR (Optimized Link State Routing) est un protocole proactif à état de lien, qui utilise un mécanisme d'inondation optimisé pour diffuser à tous les nœuds du réseau des informations partielles sur les liens.

Le trafic de contrôle dans OLSR se compose de deux types de messages: HELLO et TC. Les HELLOs sont envoyés périodiquement par un nœud pour signaler ses liens (symétriques, asymétriques ou MPR) avec les nœuds voisins, et ne sont pas relayés; accessoirement, l'échange de messages HELLO permet à chaque nœud de mémoriser des informations sur son voisinage à deux sauts, informations qui seront par la suite utilisées pour la sélection des MPRs. Les TCs sont émis périodiquement par un nœud si celui-ci a été sélectionné comme MPR, et contiennent une liste de voisins symétriques du nœud; ces messages sont diffusés dans le réseau entier. Deux autres types de messages, MID et HNA, sont émis par un nœud ayant des interfaces multiples respectivement OLSR et non-OLSR, pour annoncer la configuration de ses interfaces au réseau. Ces messages de contrôle sont encapsulés dans un paquet OLSR.

OLSR utilise un système d'inondation optimisée basée sur un sous-groupe de nœuds appelés *Relais Multipoint (MPR)*. Chaque nœud sélectionne ses MPRs parmi ses voisins symétriques de telle façon qu'un message envoyé par le nœud et répété par ses MPRs (son *MPR set*) sera reçu par tous les voisins à deux sauts du nœud en question. Chaque nœud mémorise aussi un *MPR selector set*, qui contient l'adresse de ses voisins qui l'ont sélectionné comme MPR. Les messages de contrôle sont relayés seulement par les MPRs.

A.2 Sécurité des systèmes

La sécurité d'un système inclut plusieurs problématiques telles que le contrôle d'accès, l'authentification, la confidentialité, l'intégrité, la non-répudiation et la disponibilité de service. Ces qualités sont menacées par les attaques correspondantes: accès non autorisé, usurpation d'identité, écoute passive, modification des messages, falsification des messages et Déni de Service (DoS). Les contre-mesures visant à prévenir ces attaques font souvent appel à la cryptographie, qui permet le chiffrement, la génération d'un digest et/ou la signature numérique des messages échangés à travers le système.

Ces techniques peuvent être mises en place soit au moyen de la cryptographie symétrique, avec une même clé secrète partagée pour chiffrer et pour déchiffrer, et qui utilise des fonctions de hachage pour générer des digests; soit au moyen de la cryptographie asymétrique, avec deux clés différentes (paire clé privée / clé publique) pour le chiffrement et pour le déchiffrement, et qui permet d'assigner une clé de signature différente à chaque participant. Ce dernier mécanisme nécessite la mise en place d'une Infrastructure à Clé Publique (PKI), avec dans la plupart des cas la présence d'une Autorité de Certification (CA) pour certifier qu'une certaine clé appartient bien à tel utilisateur.

A.3 Attaques contre les réseaux ad hoc

Un réseau sans fil est davantage versatile mais davantage vulnérable aux attaques qu'un réseau filaire, car les transmissions radio sont effectuées dans l'air.

Sur un réseau filaire, un intrus nécessiterait d'avoir un accès physique à une machine du réseau, ou bien de se connecter aux câbles. Dans le cas d'un réseau sans fil, l'intrus peut écouter passivement tous les messages échangés pourvu qu'il se trouve dans l'aire d'émission, en opérant en "promiscuous mode" et en utilisant un logiciel packet sniffer. Donc l'adversaire a accès au réseau et peut intercepter aisément les données transmises, sans même que l'émetteur ait connaissance de l'intrusion (par exemple, au moyen d'un ordinateur portable dans un véhicule stationné dans une rue on peut intercepter les communications échangées à l'intérieur d'un immeuble voisin). L'intrus, en étant potentiellement invisible, peut enregistrer, modifier, et ensuite retransmettre les paquets comme s'ils avaient été envoyés par un utilisateur légitime.

En outre, à cause des limitations du support, les communications peuvent facilement être perturbées; l'intrus peut effectuer cette attaque en occupant le support avec ses propres messages, ou tout simplement en perturbant les communications avec du bruit.

A.3.1 Attaques contre les MANETs au niveau du routage

Les attaques contre le protocole de routage des réseaux ad hoc peuvent avoir pour but de modifier le protocole lui-même, pour que le trafic passe par d'un nœud contrôlé par l'adversaire. Une attaque peut aussi avoir pour but d'empêcher la formation du réseau, obliger les nœuds à mémoriser des routes incorrectes, et en général perturber la topologie du réseau.

Les attaques au niveau du routage peuvent être classées dans deux catégories: génération et relayage incorrect du trafic. Nous ne considérons pas la composante des données dans le trafic, mais seulement les messages de contrôle du protocole de routage. Parfois les mêmes inconvénients ne sont pas dus à une attaque mais viennent de problèmes de fonctionnement d'un nœud, de l'épuisement des batteries, ou des interférences radio.

Géneration incorrecte du trafic

Cette catégorie inclut les attaques qui consistent en faux messages de contrôle envoyés avec l'identité d'un autre nœud (*identity spoofing*). Les conséquences sont un possible conflit d'information dans les différentes parties du réseau, dégradation des communications, nœuds non joignables, et boucles dans les parcours de routage.

Dans un protocole de routage à vecteur de distance, un nœud adversaire peut déclarer une distance de zéro pour toutes les destinations, ce qui fait que tous les nœuds autour de lui vont router leurs paquets vers le nœud adversaire. Ensuite, l'adversaire peut couper les communications dans le réseau en rejetant les paquets reçus au lieu de les faire suivre. Dans un protocole à état de lien, l'adversaire peut déclarer faussement des liens avec des nœuds distants. En conséquence, les nœuds mémorisent des fausses informations dans leurs tables de routage (*cache poisoning*).

Un adversaire peut aussi bien effectuer un Déni de Service en saturant le support avec une grosse quantité de messages en broadcast, en réduisant le débit des nœuds et, au pire, les empêchant de communiquer. L'adversaire peut aussi envoyer des messages non valables qui ont pour seul but de maintenir les nœuds actifs et d'épuiser leurs batteries.

Kuzmanovic et Knightly ont démontré l'efficacité d'une attaque DoS à longue périodicité (*shrew attack*) sur la couche transport, qui de plus n'est pas décelé par les techniques anti-DoS. En cas de congestion grave du réseau, le protocole TCP suit les périodes de Retransmission Time Out (RTO). Le flot de données (y compris les paquets DoS) déclenche le protocole de congestion TCP, donc le flux TCP entre en timeout et attend une période RTO avant d'essayer à nouveau d'envoyer un autre paquet. Si la périodicité de l'attaque est proche du RTO, les tentatives successives du flux se soldent par un échec, ce qui résulte en un débit nul.

Le *jellyfish attack* est une autre attaque DoS sur la couche transport

très rusée et difficile à déceler. Il peut être accompli de trois façons: en relayant les paquets TCP en désordre au lieu de l'ordre FIFO canonique, en rejetant les paquets pour un court laps de temps à chaque période RTO, ou en augmentant la variation de délai en retenant un paquet TCP pour une période aléatoire avant de le traiter.

Relayage incorrect du trafic

Les communications en provenance de nœuds légitimes peuvent être polluées par des nœuds malveillants. Un nœud adversaire peut éviter de relayer les messages qu'il reçoit au fin de réduire la quantité d'information disponible aux autres nœuds. Ceci a été appelé *blackhole attack* (attaque trou noir) par Hu et al., et s'agit d'un moyen simple d'effectuer un DoS. Cette attaque peut être operée sur la totalité ou une partie des paquets reçus, en rendant injoignable ou difficilement joignable le nœud destination.

Un adversaire peut aussi modifier les messages qu'il reçoit avant de les renvoyer, si un système de digest pour garantir l'intégrité n'a pas été mis en place.

Une autre attaque est le rejeu des messages: au fur et à mesure que la topologie change, les anciens messages de contrôle, quoique valables dans le passé, décrivent une configuration qui n'existe plus. Un adversaire peut enregistrer des messages de contrôle pour les rejouer plus tard, dans le but d'inclure des vieilles routes dans les mises à jour des tables de routage des nœuds. Cette attaque marche même en présence d'un système de signature ou de digest, si celui-ci n'inclut pas un estampillage temporel des messages.

Une attaque très difficile à parer est le *wormhole* (attaque trou de ver), effectué par un nœud intrus X situé à portée de transmission de deux nœuds légitimes A et B qui n'ont pas de lien entre eux. Le nœud X échange les messages entre A et B sans y ajouter son adresse dans l'entête; ceci a le résultat de créer entre A et B un lien inexistant sous le pouvoir de l'attaquant X, qui est pratiquement invisible.

Le *rushing attack* est utilisé contre les protocoles de routage à la demande; lors d'une découverte de route, l'adversaire relaye en premier son message de Route Request. Si c'est le Route Request qui parvient en premier au destinataire, la route trouvée inclura le nœud adversaire.

A.3.2 Attaques contre le protocole OLSR

Nous discutons maintenant des risques de sécurité dans OLSR. Le but n'est pas de remarquer les failles dans OLSR, car il n'a pas été conçu comme protocole sécurisé, mais de donner des exemples des risques que courent tous les protocoles à état de liens, comme OSPF.

Géneration incorrecte du trafic

Un nœud malveillant X peut envoyer des HELLOs ayant une fausse origine C. En conséquence, d'autres nœuds pourraient, en se trompant, déclarer être voisins de C à travers leurs messages HELLO et TC. En outre, le nœud X choisit ses MPRs parmi ses voisins avec l'identité de C; de ce fait, ces MPRs vont déclarer qu'il sont voisins de C. L'effect de cette attaque se traduit par des conflits des routes vers C, avec perte de connectivité.

Nous appelons *link spoofing* la signalisation d'une relation de voisinage avec des nœuds qui en fait ne sont pas des voisins. Un nœud X déclarant faussement un lien avec un nœud éloigné obtient un faux voisinage à deux sauts pour ses voisins, et donc une mauvaise selection des MPRs. Le nœud X peut aussi signaler un ensemble incomplet de voisins; les voisins ignorés pourraient éventuellement se trouver coupés du reste du réseau.

Si le nœud X envoie un TC ayant pour origine C et déclarant A comme voisin, le nœud D mémorisera faussement une relation de voisinage entre C et A. Des messages TC qui contiennent des faux liens ont aussi cet effet néfaste, et peuvent perturber la topologie du réseau.

Une autre attaque concerne l'envoi de messages MID/HNA déclarant des interfaces inexistantes, ce qui a des effets délétères envers les nœuds essayant de joindre ces interfaces.

Un nœud malveillant peut aussi générer des TCs avec une fausse origine A et un ANSN (Advertised Neighbor Sequence Number) plus élévé que celui du dernier TC envoyé par A. Tous les nœuds ignoreront donc tout message TC ultérieur de la part de A, parce qu'il porte un ANSN avec une valeur inférieure. Nous appelons ceci une *attaque ANSN*.

Relayage incorrect du trafic

Un dégât important peut être apporté au réseau, en termes de connectivité, si les messages TC ne sont pas relayés (blackhole attack). Le non-relayage des messages MID/HNA peut lui aussi engendrer des pertes d'informations dans certaines parties du réseau.

Concernant les attaques de rejeu, un TC ne peut pas être rejoué à moins d'augmenter son ANSN, engendrant ainsi une attaque ANSN.

Un wormhole peut être créé par un nœud intrus X en faisant suivre les messages de A vers B et viceversa. L'attaque commence à être efficace quand A et B sont unis par un lien symétrique; jusqu'à ce moment là, tout message TC/MID/HNA acheminé à travers le wormhole est refusé soit par A soit par B, parce que les spécifications de OLSR imposent que ces messages soient rejetés si le nœud émetteur n'est pas un voisin symétrique.

Un adversaire peut exploiter la règle d'OLSR qui spécifie qu'un nœud recevant un message en inondation MPR ne retransmet plus le message si l'envoyeur est son MPR selector. Cette attaque, que nous avons appelé *at-*

taque MPR, est produite par une retransmission illicite du message effectuée par l'attaquant.

A.4 Sécurité dans les réseaux ad hoc: mécanismes de base

Les transmissions sans fil utilisent un support partagé – l'air – qui est accessible à tout le monde. Comme il est impossible de limiter l'accès au support, la seule solution pour protéger les messages est d'utiliser la cryptographie.

A.4.1 Protection du protocole de routage

Normalement, quand on parle de la sécurisation du routage, on désire assurer l'intégrité, la non-répudiation (parfois) et la disponibilité de service. La protection des messages de routage est garantie par une signature ou un digest; ce n'est pas important de chiffrer les messages, car les informations topologiques normalement ne sont pas secrètes.

Dans la littérature il existe plusieurs protocoles de routage sécurisés par l'ajout d'une signature ou d'un digest dans les paquets de contrôle: pour exemple SRP, SLSP, SAODV, ARAN, Ariadne, SEAD, et la technique MAE. Le protocole SAR incorpore le niveau de sécurité intrinsèque d'un nœud (en ce qui concerne sa sûreté, importance, ou capacité) dans la métrique de routage pour acheminer les messages à travers des chemins considérés sûrs. D'autres protocoles ont été explicitement conçus comme défense à des attaques spécifiques, comme TIK contre le wormhole attack ou RAP contre le rushing attack. Lee et al. ont envisagé de sécuriser DSR en y ajoutant des messages de confirmation dans la découverte de route. Enfin, Buttyán et Hubaux ont proposé des mécanismes pour renforcer la disponibilité du service dans un réseau ouvert, comme le Packet Purse Model et le Packet Trade Model.

Nous avons envisagé la possibilité d'utiliser un standard très connu, IPsec, pour la protection du routage dans OLSR. Toutefois, du fait que IPsec demande qu'une association de sécurité soit déjà établie entre deux paires (ce qui n'est pas le cas dans un réseau ad hoc en formation), que la protection dans IPsec est faite à l'égard d'un paquet entier, de la difficulté de gestion d'une clé de groupe dans un réseau ad hoc, et en général des problèmes d'authentification d'un nouveau nœud qui rejoint le réseau, IPsec ne paraît pas être la solution appropriée.

L'état de l'art au sujet de la sécurisation du protocole OLSR comprend une solution pour ajouter un digest à chaque paquet, avec une vérification de la signature qui est nécessairement effectuée saut par saut. Une autre solution prévoit pour le routage un système de métrique basé sur la confiance, avec la division du réseau en différents domaines de sécurité selon la fiabilité

des nœuds qui le composent.

A.5 Le message de signature dans OLSR

Nous allons décrire notre projet de sécurisation d'OLSR, qui fait appel à l'ajout d'une signature aux messages de contrôle. Un digest, généré au moyen d'une clé symétrique partagée, peut aussi bien être utilisé à la place de la signature.

A.5.1 Spécifications du projet

La signature est calculée sur le corps et l'entête du message, et est distribuée sous la forme d'un type spécial de message, appelé SIGNATURE. Un message SIGNATURE est généré et envoyé avec tout autre message de contrôle (HELLO, TC, MID ou HNA). Il n'est pas possible de signer un paquet entier parce qu'il peut contenir des HELLOs, qui ne sont pas relayés, et donc la signature du paquet ne serait plus valable au delà du premier saut. Une solution serait celle de contrôler la signature saut par saut; toutefois, comme les messages sont relayés par inondation MPR, tout nœud qui a relayé un message incorrect pourrait en être l'émetteur, tandis qu'une authentification par message permet de déterminer aisément l'origine des fausses informations.

On identifie sans ambiguïté à quel message appartient une signature car les deux doivent se trouver dans un même paquet OLSR et sont consécutifs. Dans une version précédente, le couplage était identifié grâce au Numéro de Séquence (MSN) du message de contrôle en question et à un champ homologue dans le message de signature; cela permettait d'envoyer les messages dans un ordre quelconque, et même dans des paquets différents.

Si la taille du paquet dépasse le MTU, le message de contrôle est fragmenté et un message SIGNATURE est associé à chaque fragment. Le message SIGNATURE contient aussi une estampille temporelle, obtenue de l'horloge interne du nœud, pour éviter les attaques de rejeu; la synchronisation des horloges ne nécessite pas d'être très précise, puisque les message qui seraient des doublons peuvent être reconnus aussi par leur Numéro de Séquence (qui est enregistré dans le Duplicate Set).

Notre implantation a recours à la cryptographie asymétrique et une CA en modalité non en ligne pour assigner une paire de clés à chaque nœud participant; chaque nœud diffuse ensuite sa clé publique aux autres nœuds. On utilise les signatures Cha-Cheon, basées sur l'identité, pour les clés assignés par la CA (*clés globales*) et qui seront ensuite utilisées pour signer les clés des nœuds (*clés locales*); pour ces dernières on a choisi les signatures courtes Boneh-Lynn-Shacham.

La signature n'inclut pas les champs TTL et de compte de sauts (dans l'entête du message). Cela est dû au fait que ces deux champs sont modifiés à chaque saut du message, ce qui interférerait avec la vérification de la

signature. Malheureusement ce fait permettrait à un adversaire de relayer des messages avec un TTL modifié à 0 en restant inaperçu. Cette faille peut être résolue en ignorant le champ TTL et en considérant à sa place la valeur de l'estampille temporelle.

Cette architecture de sécurité n'est pas interopérable avec OLSR standard. En effet, un nœud dans lequel tourne OLSR sécurisé n'accepterait pas des HELLOs non signés de la part des nœuds OLSR standard; en conséquence il ne pourrait pas y avoir de lien symétrique entre les deux, et donc aucune sélection des MPRs qui est le mécanisme principal pour la diffusion des messages dans OLSR.

A.5.2 Modifications du protocole OLSR standard

Au moment de la création d'un message de contrôle, un nœud doit générer aussi un message SIGNATURE et y écrire les champs relatifs au temps et à la signature. Un nœud recevant ces messages doit retenir la SIGNATURE et vérifier si le message de contrôle est acceptable du point de vue de la signature et de son temps de création; si ces vérifications réussissent, le message de contrôle est traité. Un message de contrôle ou de SIGNATURE non valable est effacé de la Duplicate Table, pour éviter qu'un adversaire remplisse la Duplicate Table d'un nœud avec des messages non valables et empêche le nœud de traiter des messages valables qui ont le même Numéro de Séquence. Le Duplicate Set est modifié avec un nouveau champ qui prend en compte l'estampille temporelle.

A.6 Systèmes cryptographiques pour les environnements ad hoc

Génériquement, il est souhaitable que un algorithme de signature/digest ait les caractéristiques suivantes, pour pouvoir être utilisé pour sécuriser un réseau ad hoc: une signature courte, un temps de vérification court, un processus de vérification plus rapide que la signature, et une complexité limitée. Cela est dû aux limites des machines (puissance de calcul et autonomie limitées) et du support. Le respect de ces caractéristiques est lié aussi à d'autres facteurs, par exemple l'implantation d'un algorithme sur une certaine architecture. Parmi les algorithmes qui pourraient être choisis, nous citons RSA, DSA et ECNR pour la cryptographie asymétrique (signature), et HMAC-MD5 ou HMAC-SHA1 pour la cryptographie symétrique (digest).

A.6.1 La gestion des clés

L'implantation d'un système à clé publique avec une Autorité de Certification s'adapte mal à un réseau ad hoc, dont les nœuds sont indépendants et mobiles et pourraient ne pas avoir la possibilité de se connecter à la CA

en permanence. En outre, la présence d'une entité centralisée constitue une vulnérabilité qui pourrait être exploitée par un adversaire pour porter des attaques DoS. Il s'agit d'un sérieux problème qui existe aussi dans les réseaux filaires.

Il est toutefois possible de réduire le poids d'une CA centralisée au moyen de la *cryptographie à seuil*, qui permet de partager l'habilitation à générer une signature parmi un certain nombre de participants; un adversaire devrait donc compromettre plusieurs nœuds pour être capable de bouleverser le système.

Une autre alternative (PKI auto-organisée) proposée par Čapkun et al. consiste en chaînes de certificats qui connectent les participants.

Le *chiffrement basé sur l'identité (IBE)* permet d'assigner à un participant une clé publique qui est dérivée de son identité ou d'autre qualités qui lui sont propres, comme son adresse IP. Dans ce cas il n'y a plus besoin d'une Autorité de Certification. Chaque participant doit toutefois demander à un tiers de confiance (l'entité génératrice de clés ou PKG), à travers un canal sûr, la clé publique correspondante à son identité.

D'autres alternatives prévoient l'assignement des clés au démarrage du réseau, au moyen d'une méthode probabiliste, ou par échange Diffie-Hellman.

Une simple PKI pour OLSR

Nous décrivons brièvement ici une simple PKI proactive qui peut être utilisée avec OLSR. Le fonctionnement de la version réactive est analogue. Cette PKI pourvoit trois classes de nœuds:

- les *autorités de signature* dont la clé publique est connue par tout autre nœud du réseau, et qui ont la responsabilité d'enregistrer les clés publiques des autres nœuds participants et de distribuer périodiquement des certificats signés contenant la liste des clés publiques des nœuds fiables;
- les *nœuds fiables*, qui sont ceux dont la clé publique est connue et certifiée par une autorité de signature;
- les *nœuds non fiables*, qui sont ceux dont la clé publique n'est pas connue ou n'est pas certifiée par une autorité de signature; il faut remarquer qu'au démarrage du réseau tout nœud, exception faite pour les autorités de signature, est non fiable.

Pour garantir la confiance dans l'information topologique qui est distribuée dans le réseau, tout nœud doit choisir ses MPRs (et accepter d'être choisi comme MPR) parmi les seuls nœuds fiables, accepter les messages TC qui proviennent des seuls nœuds fiables, et faire suivre seulement les messages qui ont été reçus des voisins fiables. Une règle simple pour exclure les nœuds non fiables du réseau serait celle de refuser tout message envoyé par un nœud non

fiable. Toutefois, ce comportement porterait à une situation d'interblocage au moment de l'initialisation du réseau, car tout nœud est non fiable à ce moment, et donc la sélection des MPRs (et la distribution des messages dans le réseau entier) serait impossible. Pour éviter cette situation, on établit qu'un nœud accepte les messages HELLO qui proviennent d'un voisin non fiable, et que ce nœud inclut ses voisins non fiables dans ses HELLOs, avec la condition que les liens MPR soient considérés simplement comme liens symétriques. En conséquence, l'autorité de signature transmettra ses certificats à ses voisins; ces voisins, après échange de messages HELLO, accepteront les voisins à deux sauts mais ne sélectionneront pas leurs MPRs parmi eux; ensuite, l'autorité de signature choisira ses MPRs parmi ses voisins pour que sa prochaine émission de certificat rejoigne tous les voisins à deux sauts.

A.7 Estampillage temporel

Comme il a été dit précédemment, un problème des systèmes distribués est qu'il est possible de rejouer des messages même si le contrôle des signatures est mis en place. Pour prévenir ce genre d'attaques, on ajoute aux messages une estampille temporelle ou un *nonce*, qui est incluse dans le calcul de la signature. Dans OLSR, le protocole de routage peut déterminer quelle information est la plus récente en examinant le MSN (Message Sequence Number) et le ANSN (Advertised Neighbor Sequence Number) des messages; ce mécanisme est toutefois suffisant pour le fonctionnement de base mais pas pour une sécurité complète, car les deux champs sont codés sur 16 bits et les débordements avec remise à zéro peuvent être fréquents.

Pour tout message émis par un nœud, une estampille temporelle est incluse. Un nœud récepteur vérifie la validité de l'estampille temporelle, en vérifiant que sa valeur ne s'écarte de la valeur de son horloge de plus d'une petite constante.

Pour ce qui concerne le contrôle temporel des messages, il existe différentes options:

- Si une protection contre les attaques de rejeu n'est pas requise, le champ relatif à l'estampillage temporel peut être simplement ignoré.

- Une solution simple pour générer des estampille temporelles serait celle d'avoir une horloge, suffisamment précise et avec une faible dérive, embarquée dans chaque nœud; cette solution peut être implantée sous forme d'une horloge au quartz ou atomique, ou bien d'un dispositif GPS pour la transmission du temps. Dans les ordinateurs de bureau, cette horloge est l'horloge interne ou du BIOS, présentant une dérive d'environ 1 seconde par jour qui toutefois peut être réduite au moyen de corrections de la fonction du temps.

- Une implantation pour des simples estampilles temporelles consiste à écrire la valeur de l'horloge dans tout message envoyé (et signé), tandis que les nœuds récepteurs maintiennent une liste des plus grandes valeurs d'horloge reçues dans un message, pour chaque nœud émetteur. Un message, de la part d'un certain nœud, est accepté s'il porte une valeur d'horloge supérieure à la valeur déjà enregistrée pour ce nœud; dans ce cas, la valeur enregistrée est mise à jour. Ce système présente des problèmes de synchronisation si les communications entre les nœuds sont coupées pendant une certaine période.

- La solution la plus sûre consiste en une synchronisation des horloges des nœuds, solution qui toutefois fait surgir un problème d'interblocage: les estampilles temporelles sont utilisées pour l'authentification, mais une synchronisation sécurisée des horloges demande aussi une authentification. Nous avons esquissé un protocole de synchronisation pour OLSR qui s'inspire du protocole Needham-Schrœder, en utilisant la signature au lieu du chiffrement et les estampilles temporelles au lieu des nonces.

A.8 Sécurité dans les réseaux ad hoc: mécanismes avancés

Nous avons vu que les signatures dans les message protègent effectivement le réseau contre les attaques d'usurpation d'identité. Toutefois, si un adversaire a réussi à prendre le contrôle d'un nœud légitime ou à s'emparer de sa clé privée, il peut générer des messages signés correctement avec son identité; un tel nœud est appelé un *nœud compromis*. Dans ce cas, aucun nœud ne peut être considéré comme fiable, car il pourrait envoyer de faux messages de contrôle pour perturber la topologie du réseau. La question est maintenant comment s'assurer que les informations fournies par un certain nœud sont correctes.

Il est toutefois encore possible de distinguer les bonnes informations des fausses. Nous présentons dans les sections qui suivent une solution basée sur des signatures multiples, et une autre basée sur l'utilisation de la position géographique des nœuds. Une section ultérieure montre comment tout mécanisme de sécurité active peut être intégré avec un système de détection des comportements illicites.

A.9 Signatures multiples dans OLSR

Dans OLSR, comme dans tout autre protocole à état de lien, la topologie du réseau dépend de la topologie telle qu'elle était à un instant précédent. Par exemple, le nœud A sélectionne à l'instant t le nœud B comme MPR. Il est donc possible d'affirmer que à l'instant $t' = t - \Delta t$ le nœud B avait

déclaré un lien symétrique avec A, et que à l'instant $t'' = t' - \Delta t'$ le nœud A avait un lien asymétrique avec B. Tous ces liens avaient été déclarés dans des messages HELLO, qui sont le moyen par lequel les nœuds établissent les liens entre eux. En bref, la topologie ne procède pas par sauts, mais évolue avec continuité, avec une précise séquence chronologique.

Nous pouvons utiliser ce fait pour éviter que des fausses informations soient inoculées dans le réseau. Le concept de base est que tout nœud mémorise l'information concernant ses liens envoyée par ses voisins, et la réutilise comme preuve dans ses messages de contrôle successifs. Cette information est signée pour éviter les contrefaçons. Un message de contrôle envoyé par un nœud compromis ne pourra donc contenir de faux liens, parce que ces liens manquent des preuves appropriées. C'est la première fois, à notre connaissance, qu'une telle technique est proposée. Pour ce système de sécurité nous avons prévu un nouveau type de message, appelé ADVSIG, qui est toujours envoyé en couple avec un HELLO ou TC.

A.9.1 Information atomique sur l'état de lien

La quantité minimale d'information échangée sur l'état de lien, générée par le nœud A concernant le nœud B, consiste en:

- l'adresse du nœud origine A
- l'adresse du nœud annoncé B
- l'état de lien de B par rapport à A
- une estampille temporelle
- la signature de ces quatre champs, calculée par A

Les trois premiers champs sont tirés du message HELLO et de son entête, tandis que les derniers deux sont contenus dans un message ADVSIG couplé à ce HELLO. Cette information atomique est appelée un *Certificat* ou une *Preuve*, selon respectivement qu'elle est reçue comme information topologique nouvelle ou qu'elle est réutilisée pour prouver un état de lien.

Quand un nœud reçoit un HELLO avec son ADVSIG, il extrait des deux messages les informations qui le concernent (à savoir, celles où l'adresse du nœud annoncé est son adresse), et ces informations constituent donc un *Certificat*. Les Certificats sont mémorisé dans la *Certiproof Table* du nœud. Ensuite, quand le nœud envoie un HELLO ou un TC, il sélectionne dans son Certiproof Table une Preuve appropriée, qu'il inclura dans son ADVSIG couplé.

A.9.2 Preuves requises

Quand un nœud A veut déclarer un lien avec le nœud B dans un message HELLO ou TC, la preuve à fournir est construite en utilisant un HELLO et son ADVSIG couplé qui ont récemment été envoyés par B. La preuve requise est:

- une preuve que le paquet a été entendu, si A veut déclarer un lien de type ASYM_LINK avec B;
- une déclaration de ASYM_LINK ou SYM_LINK, si A veut déclarer un SYM_LINK avec B;
- une déclaration de SYM_LINK ou SYM_NEIGH, si A veut déclarer un SYM_NEIGH ou un MPR_NEIGH avec B;
- une déclaration de SYM_NEIGH ou MPR_NEIGH, si A veut déclarer B comme voisin.

A.9.3 Le protocole

Quand un nœud génère un message HELLO ou TC, il doit générer aussi un ADVSIG, en suivant ce protocole:

1. créer le HELLO/TC;
2. générer l'estampille temporelle;
3. si le message est un HELLO alors, pour chaque lien déclaré, calculer la signature du Certificat et joindre la Preuve requise appropriée;
4. sinon si le message est un TC alors joindre la Preuve requise appropriée;
5. calculer la signature;
6. envoyer le HELLO/TC et le ADVSIG.

Quand un nœud reçoit un message de contrôle, il doit suivre ces étapes:

1. identifier correctement le HELLO/TC avec son ADVSIG couplé;
2. contrôler la validité de l'estampille temporelle;
3. contrôler la validité de la signature;
4. si le message est un HELLO alors, pour chaque lien déclaré, contrôler la validité de la Preuve, et extraire le Certificat relatif au nœud lui-même le cas échéant;
5. sinon si le message est un TC alors, pour chaque voisin déclaré, contrôler la validité de la Preuve.

Une Preuve n'est valable que si elle concerne le bon nœud, si le lien inclus est correct par rapport à la preuve requise, et si l'estampille temporelle n'est pas périmée. Si une erreur survient lors d'une de ces étapes, le HELLO/TC et son ADVSIG doivent être rejetés.

A.10 Utilisation des informations sur la position des nœuds

Une information utile qui peut être ajoutée dans un message de contrôle, pour obtenir de la redondance et donc renforcer la sécurité, c'est la position géographique d'un nœud. Il existe déjà des protocoles de routage, comme DREAM, GPSR et LAR, qui utilisent cette information pour le fonctionnement de base du routage ou, comme SPAAR, pour la sécurisation du protocole. La position peut être obtenue par des dispositifs satellitaires GPS embarqués dans chaque nœud.

A.10.1 GPS-OLSR

Nous proposons une extension sécurisée pour OLSR, appelée GPS-OLSR, qui inclut dans les messages de contrôle la position géographique du nœud émetteur. Cette information est ensuite retenue par les nœuds destinations pour évaluer la véracité des informations incluses dans le même message de contrôle. Tout nœud mémorise la dernière position connue de chaque autre nœud du réseau dans sa *Position Table*.

En effet, en connaissant les positions géographiques d'un nœud émetteur S et d'un nœud récepteur R à des moments définis, en calculant la variation de leur position (qui est à son tour limitée par la vitesse maximale d'un nœud), et en prenant en compte les erreurs dans la synchronisation des horloges et dans d'autres variables, on peut calculer leur distance au moment de la transmission. Cette distance ne peut pas être supérieure à la portée maximale de transmission: si c'est le cas, le lien est probablement faux. Cela permet à un nœud d'évaluer non seulement les transmission qu'il reçoit (et de savoir si elles sont, par exemple, acheminées à travers un wormhole) mais aussi d'évaluer les déclarations de voisinage d'un autre nœud: si un nœud déclare avoir un lien avec un nœud qui est très loin, cette déclaration est fortement suspecte.

En conséquence ce protocole sécurise le réseau contre les attaques de link spoofing et wormhole. Il faut remarquer que ce mécanisme offre aussi des possibilités d'amélioration du protocole OLSR standard, telles qu'une sélection plus efficace des MPRs ou la prévision de rupture des liens.

De surcroît, l'utilisation d'une antenne directionnelle permettrait, avec des simples calculs de géométrie planaire, de savoir avec plus de précision si les informations reçues sont correctes ou fausses: un nœud peut vérifier

si le secteur d'antenne dans lequel la transmission est entendue s'accorde avec la direction vers laquelle le nœud émetteur devrait se trouver (direction obtenue en évaluant sa position relative).

Dans notre proposition d'implantation, l'information géographique est incluse dans un nouveau type de message signé appelé SIGLOC. Ce message est construit comme un message SIGNATURE avec un champ supplémentaire qui contient la position du nœud, et est envoyé avec tout HELLO ou TC.

A.11 Détection des comportements hostiles

Les protocoles sécurisés ont pour but de prévenir les attaques; en revanche, dans le cas d'une attaque avérée, les systèmes d'audit sont également importants. Ces systèmes d'audit ont pour but la détection des comportements hostiles dans le réseau, l'alerte des autres nœuds et la mise en place d'une contre-mesure pour exclure le nœud malveillant du réseau. Ces techniques peuvent être utilisées avec ou sans une infrastructure pour l'authentification des nœuds, toutefois les messages d'alerte signés évitent que l'outil de détection soit abusé par le nœud malveillant.

Les systèmes de détection actuels, comme le Watchdog/Pathrater (chien de garde / évaluateur de parcours), CONFIDANT ou Bloodhound, se basent sur l'écoute passive des transmissions pour déceler si les paquets sont correctement relayés; d'autres, comme WATCHERS, utilisent le principe de la conservation du flux; d'autres encore adoptent des paquets d'acquittement ou de test.

A.11.1 Un système pour OLSR basé sur la confiance

Le problème dans les protocoles à audit distribué c'est la difficulté à évaluer les affirmations d'un nœud qui en accuse un autre: il n'est pas possible de savoir si le premier nœud suit le protocole et le deuxième ne le suit pas, ou si le premier est malveillant et accuse faussement le deuxième dans le but de perturber le réseau. Toutefois nous pouvons supposer que, étant donné la difficulté de casser une infrastructure cryptographique, les nœuds légitimes surpassent en nombre les nœuds compromis. On peut donc utiliser cet avantage en exigeant que les alertes soient confirmées par plusieurs nœuds.

Nous proposons un protocole de détection pour OLSR qui utilise un système d'évaluation du taux de confiance des nœuds. Une *Trust Table* globale, dont tout nœud maintient une copie en sa mémoire, associe à chaque nœud une valeur numérique qui représente son niveau de confiance. Quand un nœud détecte un autre nœud qui ne respecte pas le protocole, ce premier diffuse en inondation un message d'accusation signé; s'il y a un nombre suffisant de nœuds qui envoient une accusation pour un même nœud dans le même laps de temps, les nœuds réduisent la valeur du niveau de confiance du nœud accusé. Toutefois, s'il n'y a pas assez d'accusations pendant le temps

établi, c'est le nœud accusé qui voit remonter son niveau de confiance tandis que ses dénonciateurs sont pénalisés: cela pour éviter les abus de la part d'un nœud malveillant. Le niveau de confiance de tous les nœuds est périodiquement haussé d'une valeur prédéterminée pour parer les collisions, les erreurs en transmission et les pertes physiologiques de paquets qui s'avèrent même dans un réseau dépourvu de nœuds malveillants. Une fois que le niveau de confiance d'un nœud accusé tombe à zéro, ce nœud est exclu du réseau, par effacement de son adresse dans les tables de routage.

Les comportements interdits peuvent aller de la négligence dans le relayage, à l'envoi d'un message de contrôle difforme, à une fasse signature dans le message, à une usurpation d'identité jusqu'à l'essai d'un Déni de Service par bombardement de messages; à chaque comportement est associée une réduction différente du niveau de confiance.

A.11.2 Contrôles précis sur la conservation du flux

Une mesure optionnelle, basée sur le principe de la conservation du flux, permet d'effectuer des contrôles plus précis. Le principe de conservation du flux s'énonce ainsi: "Toute donnée envoyée à un nœud et non destinée à ce nœud doît sortir du nœud". Nous pouvons détailler ce principe en observant que le nombre de paquets envoyés par un nœud Z à ses voisins, moins le nombre de paquets envoyés par les voisins à Z, doit être égal au nombre de paquets envoyés par Z et ayant Z pour origine, moins le nombre de paquets envoyés à Z destinés à Z, moins le nombre de paquets envoyés à Z et jugés acheminés incorrectement par Z. Ce contrôle est effectué par les nœuds envers tous leurs voisins. Cependant, cette technique assure la livraison du bon nombre de paquets, mais ne permet de faire aucune hypothèse sur le contenu des paquets. Une solution possible serait celle de produire une liste d'empreintes ou de filtres de Bloom sur les paquets traités.

A.12 Conclusion

Dans cette thèse nous avons étudié globalement les problèmes de sécurité dans les réseaux sans fil, plus précisément les protocoles de routage pour les réseaux ad hoc, et nous avons donné notre contribution en suggérant des solutions pour sécuriser OLSR. Ces solutions incluent en première instance l'ajout d'une signature numérique au trafic de contrôle, qui est la protection canonique contre les intrusions dans le protocole de routage.

Des techniques plus élaborées, présentées dans cette thèse, s'appuient sur la validation de l'information sur l'état de lien pour éviter que des nœuds compromis ne créent de fausses informations. Il s'agit d'un niveau avancé de protection, qui suppose qu'un adversaire est capable de générer des signatures correctes pour le trafic de contrôle qui provient de certains nœuds. Ces techniques avancées utilisent des connaissances additionnelles, telles que des

déclarations précédentes d'état de lien ou bien des données géographiques qui décrivent la position d'un nœud, pour valider l'information topologique distribuée par les nœuds dans le réseau. Le renforcement de la sécurité est aux dépens de l'overhead relatif aux messages, car ces messages de contrôle sécurisés ont une taille plus importante et impliquent des calculs plus étendus, qui doivent être effectués soit par le nœud origine soit par les nœuds récepteurs. Ceci peut s'avérer impossible pour un réseau composé de nœuds qui ont une puissance de calcul insuffisante, pour un réseau implantant une Qualité de Service qui doit garantir un haut débit, ou tout simplement pour un réseau qui ne nécessite pas une sécurité renforcée. Toutefois, on peut combiner ces techniques pour garantir une sécurité encore plus grande.

Ces techniques visent la protection de l'information concernant la topologie du réseau. Les réseaux ad hoc sont le type le plus utile et souple de réseau sans fil; pour cette raison ils sont largement utilisés dans les environnements militaires. Dans ce contexte, l'information sur la topologie a beaucoup de valeur, et le réseau doit être protégé contre des intrusions qui auraient de lourdes conséquences.

En plus des techniques de prévention déjà citées, nous avons aussi décrit brièvement une méthode pour la détection et l'élimination des comportements suspects. Cette méthode vise à déceler les nœuds qui ne respectent pas le protocole et perturbent le bon fonctionnement du réseau. Une fois que les nœuds malveillants on été identifiés, une alerte est envoyée pour informer le reste du réseau. Les autres nœuds mènent ensuite une action conjointe pour éliminer les nœuds malveillants du réseau, par exemple en les effaçant des tables de routage. Ce système de détection peut être utilisé en synergie avec les techniques de prévention.

A.12.1 Perspectives

Pendant les travaux relatifs à cette thèse de doctorat nous avons trouvé des systèmes pour sécuriser OLSR, avec des spécifications et des conditions requises différentes. Il est possible de trouver d'autres systèmes en adaptant à OLSR des techniques de sécurité qui viennent d'autres protocoles à état de lien, ou même d'autres protocoles réactifs, avec les modifications nécessaires.

Nous avons brièvement illustré les algorithmes de signature qui sont utilisés dans nos systèmes. L'étude d'algorithmes cryptographiques plus performants (du point de vue d'une signature plus courte et rapide et d'une complexité de calcul inférieure) pourrait rendre les architectures proposées pour sécuriser OLSR encore plus appropriées à la réalité d'un protocole ad hoc.

List of Figures

List of Tables

Bibliography

[1] Imad Aad, Jean-Pierre Hubaux, and Edward W. Knightly. Denial of Service resilience in ad hoc networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking (MobiCom '04)*, Philadelphia, Pennsylvania, USA, September 26–October 1 2004.

[2] Cédric Adjih, Thomas Clausen, Philippe Jacquet, Anis Laouiti, Paul Mühlethaler, and Daniele Raffo. Securing the OLSR protocol. In *Proceedings of the 2nd IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2003)*, Mahdia, Tunisia, June 25–27 2003.

[3] 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, February 2005.

[4] Cédric Adjih, Daniele Raffo, and Paul Mühlethaler. Attacks against OLSR: Distributed key management for security. In *2005 OLSR Interop and Workshop*, Ecole Polytechnique, Palaiseau, France, July 28–29 2005.

[5] J. Arkko, E. Carrara, F. Lindholm, M. Naslund, and K. Norrman. MIKEY: Multimedia Internet KEYing, August 2004. RFC 3830, Standards Track.

[6] Daniel Augot, Raghav Bhaskar, Valérie Issarny, and Daniele Sacchetti. An efficient Group Key Agreement protocol for ad hoc networks. In *Proceedings of the 1st International Workshop on Trust, Security and Privacy for Ubiquitous Computing (TSPUC 2005)*, Taormina, Italy, June 12–16 2005.

[7] Dirk Balfanz, D. K. Smetters, Paul Stewart, and H. Chi Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Proceedings of Network and Distributed System Security Symposium 2002 (NDSS '02)*, San Diego, CA, USA, February 2002.

[8] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '98)*, pages 76–84, Dallas, TX, USA, 1998. ACM Press.

[9] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In *Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '96)*, pages 1–15. Springer-Verlag, 1996.

[10] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS '93)*, pages 62–73, Fairfax, VA, USA, November 3–5 1993.

[11] Steven M. Bellovin. The security flag in the IPv4 header, April 1 2003. RFC 3514, Informational (!).

[12] Raghav Bhaskar. Group Key Agreement in ad hoc networks. Technical Report INRIA RR-4832, CODES and ARLES Projects, INRIA Rocquencourt, May 2003.

[13] Uyless Black. *Internet Security Protocols: Protecting IP Traffic.* Prentice-Hall Inc., 2000.

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

[15] L. Blunk and J. Vollbrecht. PPP Extensible Authentication Protocol (EAP), March 1998. RFC 2284, Standards Track.

[16] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil pairing. *Lecture Notes in Computer Science*, 2139:213–229, 2001.

[17] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security (Asiacrypt '01)*, pages 514–532, Gold Coast, Australia, December 9–13 2001. Springer-Verlag.

[18] Rajendra V. Boppana and Satyadeva P. Konduru. An Adaptive Distance Vector routing algorithm for mobile, ad hoc networks. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, pages 1753–1762, 2001.

[19] Kirk A. Bradley, Steven Cheung, Nick Puketza, Biswanath Mukherjee, and Ronald A. Olsson. Detecting disruptive routers: A distributed network monitoring approach. In *Proceedings of the IEEE Symposium on Research in Security and Privacy (S & P 1998)*, pages 115–124, May 1998.

[20] Sonja Buchegger and Jean-Yves Le Boudec. Performance analysis of the CONFIDANT protocol (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks). In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2002)*, EPFL Lausanne, Switzerland, June 9–11 2002.

[21] Levente Buttyán and Jean-Pierre Hubaux. Enforcing service availability in mobile ad-hoc WANs. In *Proceedings of the IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHoc 2000)*, Boston, MA, USA, August 2000.

[22] Levente Buttyán and Jean-Pierre Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5), October 2003.

[23] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *Journal of the ACM*, 51(4):557–594, July 2004.

[24] Stephen Carter and Alec Yasinsac. Secure Position Aided Ad hoc Routing. In *Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN '02)*, pages 329–334, November 4–6 2002.

[25] Jae Choon Cha and Jung Hee Cheon. An identity-based signature from gap Diffie-Hellman groups. In *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography (PKC '02)*, pages 18–30. Springer-Verlag, 2002.

[26] Steven Cheung and Karl N. Levitt. Protecting routing infrastructures from Denial of Service using cooperative intrusion detection. In *New Security Paradigms Workshop*, 1997.

[27] Thomas Clausen. The Optimized Link-State Routing Protocol version 2, July 11 2005. Internet-Draft, `draft-clausen-manet-olsrv2-00.txt`, work in progress.

[28] Thomas Clausen, Philippe Jacquet, and Laurent Viennot. Investigating the impact of partial topology in proactive MANET routing protocols. In *Proceedings of the Fifth International Symposium on Wireless Personal Multimedia Communications (WPMC 2002)*, Waikiki, Honolulu, Hawaii, USA, October 27–30 2002.

[29] Thomas Heide Clausen, Gitte Hansen, Lars Christensen, and Gerd Behrmann. The Optimized Link State Routing protocol, evaluation through experiments and simulation. In *Proceedings of the IEEE Symposium on Wireless Personal Mobile Communications*, September 2001.

[30] Thomas Clausen (ed) and Emmanuel Baccelli (ed). Securing OLSR problem statement, February 14 2005. Internet-Draft, `draft-clausen-manet-solsr-ps-00.txt`, work in progress.

[31] Thomas Clausen (ed) and Philippe Jacquet (ed). Optimized Link State Routing protocol (OLSR), October 2003. RFC 3626, Experimental.

[32] R. Coltun, D. Ferguson, and J. Moy. OSPF for IPv6, December 1999. RFC 2740, Standards Track.

[33] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Communications of the ACM*, 24(8):533–536, August 1981.

[34] D. Dhillon, T. S. Randhawa, M. Wang, and L. Lamont. Implementing a fully distributed Certificate Autorithy in an OLSR MANET. In *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC 2004)*, Atlanta, Georgia, USA, March 21–25 2004.

[35] Withfield Diffie and Martin Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, November 1976.

[36] Hans Dobbertin. The status of MD5 after a recent attack. *RSA Laboratories CryptoBytes*, 2(2), 1996.

[37] Danny Dolev, Joseph Y. Halpern, Barbara Simons, and Ray Strong. Dynamic fault-tolerant clock synchronization. *Journal of the ACM*, 42(1):143–185, 1995.

[38] Gopal Dommety and Raj Jain. Potential networking applications of Global Positioning Systems (GPS). Technical Report OSU TR-24, Department of Computer and Information Science, Ohio State University, April 1996.

[39] John R. Douceur. The Sybil attack. In *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02)*, Cambridge, MA, USA, March 7–8 2002.

[40] D. Eastlake 3rd and P. Jones. US Secure Hash Algorithm 1 (SHA1), September 2001. RFC 3174, Informational.

[41] 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 (CCS '02)*, pages 41–47, Washington, DC, USA, 2002. ACM Press.

[42] Broadband Radio Access Networks (BRAN); HIgh PErformance Radio Local Area Network (HIPERLAN) type 1; functional specification. Technical Report EN 300 652 ref. REN/BRAN-10-01, ETSI, 1998.

[43] Broadband Radio Access Networks (BRAN); HIPERLAN type 2; system overview. Technical Report TR 101 683 ref. DTR/BRAN-0023002, ETSI, 1997.

[44] Broadband Radio Access Networks (BRAN); HIgh PErformance Radio Local Area Network (HIPERLAN) type 2; requirements and architectures for wireless broadband access. Technical Report TR 101 031 ref. RTR/BRAN-0022001, ETSI, 1998.

[45] Rob Flickenger. *Building Wireless Community Networks.* O'Reilly & Associates Inc., 2003.

[46] Eran Gabber and Avishai Wool. How to prove where you are: tracking the location of customer equipment. In *Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS '98)*, pages 142–149, San Francisco, CA, USA, 1998. ACM Press.

[47] Martin Gagné. Identity-Based Encryption: a survey. *RSA Laboratories CryptoBytes*, 6(1):10–19, 2003.

[48] J. J. Garcia-Luna-Aceves, Chane L. Fullmer, Ewerton Madruga, David Beyer, and Thane Frivold. Wireless Internet gateways (WINGs). In *Proceedings of the IEEE Military Communications Conference (MILCOM '97)*, pages 1271–1276, Monterey, CA, USA, November 1997.

[49] J.J. Garcia-Luna-Aceves, Marcelo Spohn, and David Beyer. Source Tree Adaptive Routing (STAR) protocol, October 22 1999. Internet-Draft, `draft-ietf-manet-star-00.txt`, work in progress.

[50] Simson Garfinkel and Gene Spafford. *Web Security, Privacy & Commerce.* O'Reilly & Associates Inc., 2001.

[51] Simson Garfinkel, Gene Spafford, and Alan Schwartz. *Practical Unix & Internet Security.* O'Reilly & Associates Inc., 2003.

[52] Mario Gerla, Xiaoyan Hong, Li Ma, and Guangyu Pei. Landmark routing protocol (LANMAR) for large scale ad hoc networks, November 17 2002. Internet-Draft, `draft-ietf-manet-lanmar-05.txt`, work in progress.

[53] Mario Gerla, Xiaoyan Hong, and Guangyu Pei. Landmark routing for large ad hoc wireless networks. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM 2000)*, San Francisco, CA, USA, November 2000.

[54] Mario Gerla, Xiaoyan Hong, and Guangyu Pei. Fisheye State Routing protocol (FSR) for ad hoc networks, June 17 2002. Internet-Draft, `draft-ietf-manet-fsr-03.txt`, work in progress.

[55] Li Gong. A security risk depending on synchronized clocks. *ACM Operating System Review*, 26(1):49–53, 1992.

[56] Li Gong. Variations on the themes of message freshness and replay – or the difficulty of devising formal methods to analyze cryptographic protocols. In *Proceedings of the Computer Security Foundations Workshop VI*, pages 131–136. IEEE Computer Society Press, 1993.

[57] Zygmunt J. Haas. A new routing protocol for the reconfigurable wireless networks. In *Proceedings of the 6th IEEE International Conference on Universal Personal Communications (ICUPC '97)*, volume 2, pages 562–566, San Diego, CA, USA, October 1997.

[58] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. The Interzone Routing Protocol (IERP) for ad hoc networks, July 2002. Internet-Draft, `draft-ietf-manet-zone-ierp-02.txt`, work in progress.

[59] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. The Intrazone Routing Protocol (IARP) for ad hoc networks, July 2002. Internet-Draft, `draft-ietf-manet-zone-iarp-02.txt`, work in progress.

[60] Andreas Hafslund, Andreas Tønnesen, Roar Bjørgum Rotvik, Jon Andersson, and Øivind Kure. Secure extension to the OLSR protocol. In *2004 OLSR Interop and Workshop*, San Diego, CA, USA, August 6–7 2004.

[61] D. Harkins and D. Carrel. The Internet Key Exchange (IKE), November 1998. RFC 2409, Standards Track.

[62] H. Harney and C. Muckenhirn. Group Key Management Protocol (GKMP) specification, July 1997. RFC 2093, Experimental.

[63] Jonathan S. Held and John R. Bowers. *Securing E-Business Applications and Communications.* Auerbach Publications, 2001.

[64] Fan Hong, Liang Hong, and Cai Fu. Secure OLSR. In *Proceedings of the 19th IEEE International Conference on Advanced Information Networking and Applications (AINA '05)*, Tamkang University, Taiwan, March 28–30 2005.

[65] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002)*, pages 3–13, Calicoon, NY, USA, June 2002.

[66] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the 8th Annual ACM International Conference on Mobile Computing and Networking (MobiCom '02)*, September 2002.

[67] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Packet leashes: A defense against wormhole attacks in wireless ad hoc networks. In *Proceedings of the Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2003)*, San Francisco, CA, USA, April 2003.

[68] 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, San Diego, CA, USA, 2003. ACM Press.

[69] Jean-Pierre Hubaux, Levente Buttyán, and Srđan Čapkun. The quest for security in mobile ad hoc networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, 2001.

[70] John R. Hughes, Tuomas Aura, and Matt Bishop. Using conservation of flow as a security mechanism in network protocols. In *Proceedings of the IEEE Symposium on Research in Security and Privacy (S & P 2000)*, pages 131–132, May 14–17 2000.

[71] Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-speed physical layer in the 5 GHz band. Technical Report IEEE Std 802-11a-1999(R2003), IEEE, 2003. ISO/IEC 8802-11:1999/Amd 1:2000(E).

[72] Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-speed physical layer extension in the 2.4 GHz band. Technical Report IEEE Std 802.11b-1999 (R2003), IEEE, 2003.

[73] Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 4: Further higher data rate extension in the 2.4 GHz band. Technical Report IEEE Std 802.11g-2003, IEEE, 2003.

[74] Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Amendment 6: Medium Access Control (MAC) security enhancements. Technical Report IEEE Std 802.11i-2004, IEEE, 2004.

[75] Part 16: Air interface for fixed broadband wireless access systems. Technical Report IEEE Std 802.16-2001, IEEE, 2002.

[76] Standard specifications for public key cryptography. Technical Report IEEE 1363-2000, IEEE, 2000. `http://grouper.ieee.org/groups/1363/P1363`.

[77] Standard specifications for public key cryptography - amendment 1: Additional techniques. Technical Report IEEE 1363A-20004, IEEE, 2004. `http://grouper.ieee.org/groups/1363/P1363a`.

[78] Shuichi Isida, Eriko Ando, and Yasuko Fukuzawa. Secure routing functions for OLSR protocol. In *2005 OLSR Interop and Workshop*, Ecole Polytechnique, Palaiseau, France, July 28–29 2005.

[79] Philippe Jacquet, Paul Mühlethaler, Thomas Clausen, Anis Laouiti, Amir Qayyum, and Laurent Viennot. Optimized Link State Routing protocol for ad hoc networks. In *Proceedings of the IEEE International Multitopic Conference (INMIC 2001)*, Pakistan, 2001.

[80] Philippe Jacquet, Pascale Minet, Anis Laouiti, Laurent Viennot, Thomas Clausen, and Cédric Adjih. Multicast Optimized Link State Routing, November 2001. Internet-Draft, `draft-ietf-manet-olsr-molsr-01.txt`, work in progress.

[81] Mingliang Jiang, Jinyang Li, and Y. C. Tay. Cluster Based Routing Protocol (CBRP), August 14 1999. Internet-Draft, `draft-ietf-manet-cbrp-spec-01.txt`, work in progress.

[82] David B. Johnson and David A. Maltz. Dynamic Source Routing in ad hoc wireless networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353, pages 153–181. Kluwer Academic Publishers, 1996.

[83] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The Dynamic Source Routing protocol for mobile ad hoc networks (DSR), July 19 2004. Internet-Draft, `draft-ietf-manet-dsr-10.txt`, work in progress.

[84] Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for wireless networks. In *Mobile Computing and Networking*, pages 243–254, 2000.

[85] S. Kent and R. Atkinson. IP Authentication Header, November 1998. RFC 2402, Standards Track.

[86] S. Kent and R. Atkinson. IP Encapsulating Security Payload (ESP), November 1998. RFC 2406, Standards Track.

[87] S. Kent and R. Atkinson. Security architecture for the Internet Protocol, November 1998. RFC 2401, Standards Track.

[88] Young-Bae Ko and Nitin H. Vaidya. Location-Aided Routing (LAR) in mobile ad hoc networks. In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '98)*, pages 66–75, Dallas, TX, USA, 1998. ACM Press.

[89] Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels. Technical Report CACR 2005-08, University of Waterloo, Waterloo, Ontario, Canada, 2005.

[90] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu, and Lixia Zhang. Providing robust and ubiquitous security support for mobile ad-hoc networks. In *9th IEEE International Conference on Network Protocols (ICNP 2001)*, pages 251–260, 2001.

[91] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-hashing for message authentication, February 1997. RFC 2104, Informational.

[92] Aleksandar Kuzmanovic and Edward W. Knightly. Low-rate TCP-targeted Denial of Service attacks (The shrew vs. the mice and elephants). In *Proceedings of the 2003 Conference of the Special Interest Group on Data Communication (SIGCOMM '03)*, pages 75–86, Karlsruhe, Germany, 2003. ACM Press.

[93] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.

[94] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[95] Anis Laouiti. *Unicast et Multicast dans les réseaux ad hoc sans fil.* PhD thesis, Université de Versailles Saint-Quentin-en-Yvelines, 2002.

[96] Seungjoon Lee, Bohyung Han, and Minho Shin. Robust routing in wireless ad hoc networks. In *2002 International Conference on Parallel Processing Workshops (ICPPW '02)*, Vancouver, Canada, August 18–21 2002.

[97] Donggang Liu and Peng Ning. Establishing pairwise keys in distributed sensor networks. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS '03)*, pages 52–61, Washington, DC, USA, 2003. ACM Press.

[98] Pete Loshin (compiler). *Big Book of IPsec RFCs*. Morgan Kaufmann Publishers, 2000.

[99] Gavin Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 1055, pages 147–166. Springer-Verlag, 1996.

[100] Haiyun Luo, Petros Zerfos, Jiejun Kong, Songwu Lu, and Lixia Zhang. Self-securing ad hoc wireless networks. In *Proceedings of the 7th IEEE Symposium on Computers and Communications (ISCC '02)*, 2002.

[101] Ben Lynn. Authenticated Identity-Based Encryption. Cryptology ePrint Archive, Report 2002/072, June 4 2002.

[102] Davor Males and Guy Pujolle. *Wi-Fi par la pratique*. Groupe Eyrolles, 2002.

[103] John Marshall. An analysis of SRP for mobile ad hoc networks. In *Proceedings of the 2002 International Multiconference in Computer Science*, Las Vegas, USA, August 18–21 2002.

[104] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. *Mobile Computing and Networking*, pages 255–265, 2000.

[105] D. Maughan, M. Schertler, M. Schneider, and J. Turner. Internet Security Association and Key Management Protocol (ISAKMP), November 1998. RFC 2408, Standards Track.

[106] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 2001. `http://www.cacr.math.uwaterloo.ca/hac`.

[107] Ralph C. Merkle. Protocols for public key cryptosystems. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, USA, April 14–16 1980.

[108] Paul Mühlethaler. *802.11 et les réseaux sans fil*. Groupe Eyrolles, 2002.

[109] Ondrej Mikle. Practical attacks on digital signatures using MD5 message digest. Cryptology ePrint Archive, Report 2004/356, 2004.

[110] J. Moy. OSPF version 2, April 1998. RFC 2328, Standards Track.

[111] Shree Murthy and J. J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mobile Networks and Applications*, 1(2):183–197, 1996.

[112] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[113] Secure Hash Signature Standard (SHS). Technical Report FIPS PUB 180-2, NIST, August 1 2002.

[114] Digital Signature Standard (DSS). Technical Report FIPS PUB 186-2, NIST, January 27 2000.

[115] R. Ogier, F. Templin, and M. Lewis. Topology dissemination Based on Reverse-Path Forwarding (TBRPF), February 2004. RFC 3684, Experimental.

[116] 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)*, San Antonio, TX, USA, January 27–31 2002.

[117] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure link state routing for mobile ad hoc networks. In *Proceedings of the 2003 International Symposium on Applications and the Internet (SAINT '03)*, Orlando, FL, USA, January 28 2003.

[118] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye State Routing: A routing scheme for ad hoc wireless networks. In *Proceedings of the IEEE International Conference on Communications (ICC 2000)*, pages 70–74, New Orleans, LA, USA, June 2000.

[119] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-demand Distance Vector (AODV) routing, July 2003. RFC 3561, Experimental.

[120] Charles E. Perkins and Pravin Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, London, United Kingdom, 1994. ACM Press.

[121] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computer Systems and Applications*, New Orleans, LA, USA, February 25–26 1999.

[122] Radia Perlman. *Network layer protocols with Byzantine robustness.* PhD thesis, Massachussets Institute of Technology, 1988.

[123] Adrian Perrig. The BiBa one-time signature and broadcast authentication protocol. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS '01)*, pages 28–37, 2001.

[124] Adrian Perrig, Ran Canetti, Dawn Song, and J. D. Tygar. Efficient and secure source authentication for multicast. In *Proceedings of the Network and Distributed System Security Symposium (NDSS '01)*, pages 35–46, February 2001.

[125] Adrian Perrig, Ran Canetti, Doug Tygar, and Dawn Song. Efficient authentication and signature of multicast streams over lossy channels. In *Proceedings of the IEEE Symposium on Research in Security and Privacy (S & P 2000)*, pages 56–73, May 14–17 2000.

[126] Raymond L. Pickholtz, Donald L. Schilling, and Laurence B. Milstein. Theory of spread spectrum communications – a tutorial. *IEEE Transactions on Communications*, 30(5):855–884, May 1982.

[127] Ricardo Staciarini Puttini. *A security model for mobile ad hoc networks.* PhD thesis, University of Brasilia, 2004.

[128] Ricardo Staciarini Puttini, Ludovic Me, and Rafael Timóteo de Sousa. Certification and authentication services for securing MANET routing protocols. In *Proceedings of the 5th IFIP TC6 International Conference on Mobile and Wireless Communications Networks*, Singapore, October 2003.

[129] Amir Qayyum, Laurent Viennot, and Anis Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Technical Report INRIA RR-3898, HIPERCOM Project, INRIA Rocquencourt, 2000.

[130] Daniele Raffo, Cédric Adjih, Thomas Clausen, and Paul Mühlethaler. An advanced signature system for OLSR. In *Proceedings of the 2004 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, pages 10–16, Washington, DC, USA, October 25 2004. ACM Press.

[131] Daniele Raffo, Cédric Adjih, Thomas Clausen, and Paul Mühlethaler. OLSR with GPS information. In *Proceedings of the 2004 Internet Conference (IC 2004)*, Tsukuba, Japan, October 28–29 2004.

[132] Daniele Raffo, Cédric Adjih, Thomas Clausen, and Paul Mühlethaler. Securing OLSR using node locations. In *Proceedings of 2005 Euro-*

pean Wireless (EW 2005), pages 437–443, Nicosia, Cyprus, April 10–13 2005.

[133] Leonid Reyzin and Natan Reyzin. Better than BiBa: Short one-time signatures with fast signing and verifying. In *Proceedings of the 7th Australian Conference on Information Security and Privacy*, pages 144–153. Springer-Verlag, 2002.

[134] R. Rivest. The MD5 Message-Digest algorithm, April 1992. RFC 1321.

[135] Ronald Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[136] RSA cryptography standard. Technical Report PKCS #1 v2.1, RSA Laboratories, June 14 2002. `ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-1/pkcs-1v2-1.pdf`.

[137] 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 the 10th IEEE International Conference on Network Protocols (ICNP '02)*, pages 78–89. IEEE Computer Society, 2002.

[138] Andreas Savvides, Chih-Chieh Han, and Mani B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, pages 166–179, Rome, Italy, July 16–21 2001. ACM Press.

[139] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, 1995.

[140] Bruce Schneier. *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, 2000.

[141] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[142] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO '84 on Advances in Cryptology*, pages 47–53, Santa Barbara, CA, USA, 1984. Springer-Verlag New York, Inc.

[143] Victor Shoup. Practical threshold signatures. In *Proceedings of Eurocrypt 2000*, pages 207–220, 2000.

[144] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science*, 1999.

[145] Jennifer G. Steiner, Clifford Neuman, and Jeffrey I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the Usenix Winter Conference*, pages 191–202, Berkeley, CA, USA, February 1988.

[146] Michael Steiner, Gene Tsudik, and Michael Waidner. Diffie-Hellman key distribution extended to group communication. In *Proceedings of the 3rd ACM Conference on Computer and Communications Security (CCS '96)*, pages 31–37, New Delhi, India, 1996. ACM Press.

[147] Michael Steiner, Gene Tsudik, and Michael Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8):769–780, 2000.

[148] Stephen A. Thomas. *SSL & TLS Essentials: Securing the Web*. John Wiley & Sons, 2000.

[149] Fouad A. Tobagi and Leonard Kleinrock. Packet switching in radio channels: Part II – the hidden terminal problem in Carrier Sense Multiple-Access and the busy-tone solution. *IEEE Transactions on Communications*, 23(12):1417–1433, December 1975.

[150] Data sheet and specifications for Thunderbolt GPS disciplined clock. Technical report, Trimble Navigation Limited, Sunnyvale, CA, USA, 2000. `http://www.trimble.com`.

[151] Srđan Čapkun, Levente Buttyán, and Jean-Pierre Hubaux. Self-organized public-key management for mobile ad hoc networks. In *Proceedings of the ACM International Workshop on Wireless Security (WiSe)*, 2002.

[152] Srđan Čapkun, Levente Buttyán, and Jean-Pierre Hubaux. Small worlds in security systems: an analysis of the PGP certificate graph. In *Proceedings of the 2002 Workshop on New Security Paradigms*, pages 28–35, Virginia Beach, Virginia, USA, 2002. ACM Press.

[153] Srđan Čapkun, Maher Hamdi, and Jean-Pierre Hubaux. GPS-free positioning in mobile ad hoc networks. In *Proceedings of the Hawaii International Conference on System Sciences (HICSS-34)*, Maui, Hawaii, USA, January 3–6 2001.

[154] Srđan Čapkun, Jean-Pierre Hubaux, and Markus Jacobsson. Secure and privacy-preserving communication in hybrid ad hoc networks. Technical Report IC/2004/10, Swiss Federal Institute of Technology Lausanne (EPFL), Lausanne, Switzerland and RSA Laboratories, Bedford, MA, USA, 2004.

[155] John R. Vig. Introduction to quartz frequency standards. Technical Report SLCET-TR-92-1, Army Research Laboratory, Electronics and Power Sources Directorate, Fort Monmouth, NJ, USA, October 1992.

[156] Xiaoyun Wang and Hongbo Yu. How to break MD5 and other hash functions. In *Proceedings of Eurocrypt 2005*, Aarhus, Denmark, May 22–26 2005.

[157] Brian Weis. The use of RSA signatures within ESP and AH, October 2004. Internet-Draft, `draft-ietf-msec-ipsec-signatures-02.txt`, work in progress.

[158] Eli Winjum, Anne Marie Hegland, Pål Spilling, and Øivind Kure. A performance evaluation of security schemes proposed for the OLSR protocol. In *Proceedings of the IEEE Military Communications Conference (MILCOM 2005)* (to appear), Atlantic City, NJ, USA, October 17–21 2005.

[159] Eli Winjum, Øivind Kure, and Pål Spilling. Trust metric routing in mobile wireless ad hoc networks. In *Proceedings of World Wireless Congress 2004*, San Francisco, CA, USA, May 25–28 2004.

[160] Eli Winjum, Pål Spilling, and Øivind Kure. Trust metric routing to regulate routing cooperation in mobile wireless ad hoc networks. In *Proceedings of 2005 European Wireless (EW 2005)*, pages 399–406, Nicosia, Cyprus, April 10–13 2005.

[161] Alec Yasinsac, Vikram Thakur, Stephen Carter, and Ilkay Cubukcu. A family of protocols for group key generation in ad hoc networks. In *Proceedings of the IASTED International Conference on Communications and Computer Networks (CCN '02)*, pages 183–187, November 4–6 2002.

[162] Seung Yi and Robin Kravets. Practical PKI for ad hoc wireless networks. Technical Report UIUCDCS-R-2002-2273 UILU-ENG-2002-1717, University of Illinois at Urbana-Champaign, USA, August 2001.

[163] Seung Yi, Prasad Naldurg, and Robin Kravets. Security-aware ad-hoc routing for wireless networks. In *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc 2001)*, Long Beach, CA, USA, October 2001.

[164] Manel Guerrero Zapata. Secure Ad hoc On-demand Distance Vector (SAODV) routing, March 17 2005. Internet-Draft, `draft-guerrero-manet-saodv-03.txt`, work in progress.

[165] Lidong Zhou and Zygmunt J. Haas. Securing ad hoc networks. *IEEE Network*, 13(6):24–30, 1999.

[166] Lidong Zhou, Fred B. Schneider, and Robbert Van Renesse. COCA: A secure distributed online certification authority. *ACM Transactions on Computer Systems*, 20(4):329–368, 2002.

[167] Philip Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995. `http://www.pgpi.org`.