INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Securing the OLSR routing protocol with or without compromised nodes in the network

Cédric Adjih, Thomas Clausen, Anis Laouiti, Paul Muhlethaler, Daniele Raffo

N° 5494

Février 2005

Thème COM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--5494--FR+ENG

Securing the OLSR routing protocol with or without compromised nodes in the network *

Cédric Adjih, Thomas Clausen, Anis Laouiti, Paul Muhlethaler, Daniele Raffo

Thème COM — Systèmes communicants
Projet HIPERCOM

Rapport de recherche n° 5494 — Février 2005 — 55 pages

Abstract: The primary issue with respect to securing Mobile Ad hoc NETworks (MANETs) is that of ensuring network integrity even when the network is subject to attacks to break its connectivity. In this research report, we study how to secure the OLSR routing protocol [8]. We first analyse the attacks that can be launched against the network integrity. We then present mechanisms for ensuring that only "trusted" nodes are admitted into the network and, subsequently, are the only nodes used to forward traffic. We also present mechanisms for detecting and dealing with scenarios where "trusted" nodes have become compromised.

Key-words: Ad hoc network, attacks, routing protocol, connectivity, signature, time-stamps, replay.

* This work has been supported by a DGA (Délégation Générale pour l'Armement) contract with the CELAR (Centre d'ELectronique de l'ARmement)

Sécuriser le protocole de routage OLSR avec ou sans noeuds compromis dans le réseau

Résumé : Le premier problème pour sécuriser les réseaux mobiles ad hoc (MANETs) est celui du maintien de l'intégrité du réseau en présence d'attaques contre la connectivité du réseau. Dans ce rapport de recherche nous étudions comment sécuriser le protocole de routage OLSR [8]. D'abord nous analysons les attaques qui peuvent être lancées contre l'intégrité du réseau. Nous présentons ensuite des mécanismes pour assurer que seulement les noeuds "sûrs" sont admis dans le réseau et, en conséquence, ce sont les seuls noeuds utilisés pour relayer le traffic. Nous présentons aussi des mécanismes pour détecter et répondre aux cas où des noeud "sûrs" deviennent compromis.

Mots-clés : Réseau ad hoc, attaques, protocole de routage, connectivité, signature, estampille temporelle, re-jeu.

1 Introduction

A Mobile Ad-hoc NETwork (MANET) is a collection of nodes which are able to connect on a wireless medium forming an arbitrary and dynamic network. Implicitly herein is the ability for the network topology to change over time as links in the network appear and disappear. In order to enable communication between any two nodes in such a MANET, a routing protocol is employed.

Currently, two complimentary classes of routing protocols exist in the MANET world. Reactive protocols acquire routes on demand through flooding a "route request" and receiving a "route reply" (typically signaling the path taken by the route request to arrive at the destination node). I.e. the required parts of the topology graph are constructed in a node only when needed for data traffic communication. Reactive MANET routing protocols include AODV [22] and DSR [15]. The other class of MANET routing protocols is proactive, i.e. the routing protocol ensures that all nodes at all times have sufficient topological information to construct routes to all destinations in the network. This is achieved through periodic message exchange. Proactive MANET routing protocols include OLSR [8] and TBRPF [21].

1.1 Security Issues

A significant issue in the ad-hoc domain is that of the integrity of the network itself. AODV, DSR, OLSR and TBRPF allow, according to their specifications, any node to participate in the network - the assumption being that all nodes are behaving well and welcome. If this assumption fails - then the network may be subject to malicious nodes, and the integrity of the network fails.

An orthogonal security issue is that of maintaining confidentiality and integrity of the data being exchanged between communications endpoints in the network (e.g. between a mail server and a mail client). The task of ensuring end-to-end security of data communications in MANETs is equivalent to that of securing end-to-end security in traditional wired networks. Many studies have been carried out to solve this problem. One widespread solution is to create a virtual private network (VPN) in a tunnel between the two communicating nodes. IPSec is a general security architecture which allows such VPNs to be built between two communicating nodes. Despite its generality, IPSec is not designed to solve the first issue related to the integrity of the network itself. However IPSec defines a huge number of protocols and mechanisms. The reuse or adaptation of this material to secure the integrity of the network is presented briefly in an annex of this research report. The security issue of maintaining confidentiality and integrity of the data being exchanged between communications is not considered in this research report.

The primary issue with respect to securing MANET routing protocols is thus that of ensuring network integrity, even in the presence of malicious nodes. Security extensions to the reactive protocols AODV and DSR exist, in the form of SAODV [10] and Ariadne [12]. Assuming that a mechanism for key distribution is in place, these extensions employ digital signatures on the route request and route reply messages. The basic principle being that each node verifies the signature of a message and - if valid - processes the message.

In this research report, we will investigate the issues of security in the OLSR protocol. We will present mechanisms for ensuring that only "trusted" nodes are admitted into the network (and, subsequently, are the only nodes used for forwarding traffic), as well as mechanisms for detecting and dealing with the situation where a trusted node has become compromised. The solution we present will rely on time-stamps in order to counter potential "replay attacks".

1.2 Outline

The remainder of this paper is organized as follows: section 2 presents OLSR in sufficient detail to devise security mechanisms which will be integrated with the protocol. Section 3 deals with vulnerabilities of OLSR. Section 4 will summarize the requirements and describes basic mechanisms for securing OLSR. Section 5 addresses the issue of securing OLSR when nodes have been compromised (i.e. they are behaving contrary to the protocol specification although they are thought to behave accordingly). Section 6 studies how the knowledge of node locations can be used to counter a specific attack called relay attack.

There are four annexes in this research report. The first describes time-stamp generation. The second one describes a general security architecture for OLSR with a centralized PKI (Public Key Infrastructure). The third is a brief overview of signing algorithms. The fourth overviews the reuse and/or adaptation of the IPSec framework within the mechanisms we have defined to secure OLSR.

2 The Optimized Link State Routing Protocol

The Optimized Link State Routing protocol (OLSR) [6], [8] is a proactive link state routing protocol, designed specifically for mobile ad-hoc networks. OLSR employs an optimized flooding mechanism to diffuse link-state information, and diffuses only a partial link-state to all nodes in the network.

In this section, we describe the elements of OLSR required to investigate security issues. A complete description of OLSR can be found in [8].

2.1 OLSR Control Traffic

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

2.1.1 HELLO messages

HELLO messages are emitted periodically by a node, encoding its own address as well as three lists: a list of neighbors from which control traffic has been heard (but where bi-directionality is not yet confirmed), a list of neighbor nodes with which bidirectional communication has been established, and a list of neighbor nodes which have been selected to act as an MPR for the originator of the HELLO message. HELLO messages are exchanged only between neighbor nodes.

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, it is confirmed that bi-directional communication is possible between the originator and the recipient of the HELLO message.

In addition to information about neighbor nodes, the periodic exchange 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 explicitly utilized for MPR optimization - the core optimization of OLSR, described in section 2.1.3.

2.1.2 TC messages

Like HELLO messages, TC messages are emitted periodically. The purpose of a TC message is to diffuse link-state information to the entire network. Thus, a TC message contains a set of bi-directional links between a node and a subset of its neighbors. For a discussion on selecting which neighbors should be included in the TC messages in order to provide sufficient topology information, refer to [8] and [7]. TC messages are diffused to the entire network, employing the MPR optimization described in section 2.1.3.

2.1.3 Multipoint Relay Selection and Signaling

The core optimization in OLSR is that of Multipoint Relays (MPRs). The concept is as follows: each node must select MPRs from among its 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. 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: a link-status of "MPR" specifies that the link between the originator of the HELLO message and the listed address is symmetric - and that the node with the included address is selected as an MPR by the originator.

Thus, each node maintains an *MPR selector set*, describing the set of nodes which have selected it as an MPR. Upon receiving an OLSR control message, a node consults its MPR selector set to determine whether the message is to be retransmitted: if the last-hop of the control message is an MPR selector, then the message is to be retransmitted, otherwise it is not retransmitted. Figure 1 shows a node with neighbors and 2-hop neighbors. In order to achieve a network-wide broadcast, it suffices that a broadcast transmission be repeated by a subset of the neighbors. This subset is made up from the MPR-set of the node. For further information, including an efficient heuristic for computing the MPR set of a node, refer to [23].

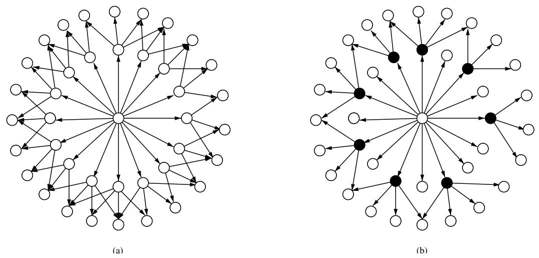

Figure 1: Two hop neighbors and "multipoint relays" (the solid circles) of a node. (a) illustrates the situation where all neighbors retransmit a broadcast, (b) illustrates where only the MPRs of a node retransmit the broadcast.

2.2 OLSR Message Format and Packets

OLSR control messages are communicated using a "transport protocol" defined by a general packet format containing individual control messages, as well as rules governing the processing of such packets and messages. In this section, we outline this transport protocol. The purpose is to outline how security extensions can be easily included, and to understand the mechanisms under which the security extensions must be designed.

The OLSR packet format is given in figure 2.

While messages may potentially be intended to be broadcasted to the entire network (e.g. a TC message), packets are transmitted only between neighbor nodes. The unit of information subject to being forwarded is "messages". The common packet format allows individual messages to be piggybacked and transmitted together in one emission (MTU-size allowing). I.e. TC and HELLO messages may be emitted together, however they are processed and forwarded differently in each node (HELLO messages are not forwarded whereas TC messages are).

It is important to notice, that an individual OLSR control message can be identified by its Originator Address and Message Sequence Number - both from the message header. Hence, disregarding issues of wraparound of the Message Sequence Number, it is possible to uniquely refer to a specific control message in the network. This will become of importance when discussing message signatures.

Further details on OLSR packet and message formats can be found in [8].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 5494 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Packet Length         |    Packet Sequence Number     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Message Type |     Vtime     |         Message Size          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Originator Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time To Live |   Hop Count   |    Message Sequence Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                            MESSAGE                            :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Message Type |     Vtime     |         Message Size          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Originator Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time To Live |   Hop Count   |    Message Sequence Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                            MESSAGE                            :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
               (etc)
```

Figure 2: Generic OLSR packet format. Each packet encapsulates several control messages into one transmission.

3 Vulnerabilities

In this section, we discuss various security vulnerabilities in proactive routing protocols for ad-hoc networks. Although we specifically enumerate vulnerabilities in OLSR, it should be stressed that this section does not emphasize "flaws" in the OLSR protocol. Rather, the vulnerabilities are instances of what all proactive routing protocols are subject to.

When an ad-hoc network is operating under a proactive routing protocol, each node has two different (but related) responsibilities. Firstly, each node must correctly *generate* routing protocol control traffic, conforming to the protocol specification. Secondly, each node is responsible for *forwarding* routing protocol control traffic on behalf of other nodes in the network. Thus incorrect behavior of a node can result from either a node generating incorrect control messages or from incorrect relaying of control traffic from other nodes.

Correctly generating and forwarding control traffic can be considered as a criterion for having a *correctly functioning routing*. I.e. that the routing protocol is able to consistently provide a correct view of the network topology in each network node. This assumption implies that all the nodes in the network correctly implement the routing protocol - and specifically that each node correctly processes and emits control traffic. Notice, that this in and by itself is not sufficient to ensure that data packets are being correctly routed in the network. Indeed, independently of whether the routing protocol is proactive or reactive, a misbehaving node may generate, process and relay control traffic correctly while not actually performing data traffic forwarding.

In the remainder of this section, we investigate how these incorrect behaviors may appear in OLSR. We note, that although we employ OLSR for the purposes of our investigations, much of the following is equally applicable to other proactive routing protocols.

3.1 Jamming

One vulnerability, common to all routing protocols operating a wireless ad-hoc network, is that of "jamming" - i.e. that a node generates massive amounts of interfering radio transmissions, which will prevent legitimate traffic (e.g. control traffic for the routing protocol as well as data traffic) on part of a network. This vulnerability cannot be dealt with at the routing protocol level (if at all), leaving the network unable to maintain connectivity. The result of jamming is somewhat similar to that of network overload: a sufficiently significant amount of routing protocol control traffic is lost, preventing routes from being constructed in the network. In this study, we will not consider a networks resistance against jamming or traffic overload.

3.2 Incorrect Message Generation

OLSR basically employs two different kinds of control traffic: HELLO messages and TC messages. In this section, we describe how a non-conforming node may affect network connectivity through the incorrect generation of HELLO and TC messages.

In general, we observe that with respect to control traffic generation, a node may misbehave in two different ways: either by generating control traffic "pretending" to be another node (i.e. *Identity Spoofing*) or by advertising incorrect information (links) in the control messages (i.e. *Link Spoofing*).

3.2.1 Incorrect HELLO Message Generation

In terms of HELLO messages, *Identity Spoofing* implies that a node sends HELLO messages, pretending to have the identity of another node. E.g. node X sends HELLO messages, with the originator address set to that of node A, as illustrated in figure 3. This may result in the network containing conflicting routes to node A. Specifically, node X will choose MPRs from among its neighbors, signaling this selection pretending to have the identity of node A. The MPRs will, subsequently, advertise that they can provide "last hop" to node A in their TC messages. Conflicting routes to node A, with possible loops, may result from this.

Similarly, *Link Spoofing* implies that a node sends HELLO messages, signaling an incorrect set of neighbors. This may take either of two forms: if the set is incomplete, i.e. a node "ignores" some neighbors, the network may be without connectivity to these "ignored" neighbors.

Alternatively, an intruder advertising a neighbor-relationship to non-present nodes may cause inaccurate MPR selection with the result that some nodes may not be reachable in the network. In figure 4, X pretends to be a neighbor of node B. Thus D can choose as its MPR set nodes X and E (smallest MPR set with this two-hop neighborhood). TC messages from F will not be delivered to B. Should X operate correctly, D would have to choose as its MPR set nodes X,C and E. TC flooding will work correctly.

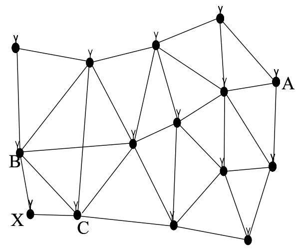

Figure 3: *Identity Spoofing* of HELLO messages: node X assumes the identity of node A to send HELLO messages. Nodes B and C may, subsequently, announce reachability to node A through their TC messages.

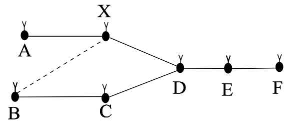

Figure 4: Wrong information about a node neighborhood may cause inaccurate MPR selection and incomplete flooding

3.2.2 Incorrect TC Message Generation

As for HELLO messages, *Identity Spoofing* with respect to TC messages implies that a node sends TC messages, pretending to have the identity of another node. Effectively, this implies *link spoofing* since a node assuming the identity of another node effectively advertises incorrect links to the network.

Similarly, *Link Spoofing* implies that a node sends TC messages, advertising an incorrect set of links. This may take either of two forms: if the set is incomplete, i.e. a node “ignores” links to some nodes in its MPR selector set, the network may be without connectivity to these “ignored” neighbors - as well as to neighbors which are reachable only through the “ignored” neighbors. A node may also include non-existing links (i.e. links to non-neighbor nodes) in a TC message. This is illustrated in figure 5.

Link spoofing in TC messages may result in routing loops and conflicting routes in the network.

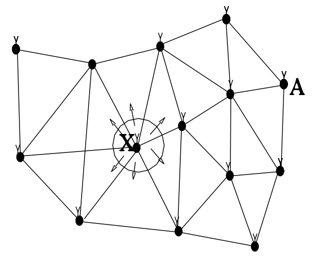

Figure 5: Node X generates incorrect TC messages, e.g. advertising a link between node X and node A.

3.2.3 Incorrect traffic Generation using correct messages previously sent: replay or relay attack

An intruder node can reuse already generated messages in the network. We first consider generated messages within an average timescale (more than a few seconds); we have incorrect messages generated by replaying old and most probably obsolete control messages.

Another possible attack is when a control traffic from one region of the network is recorded and, within a small timescale (less than a few seconds), replayed in a different region. This may, for example, happen when two nodes collaborate on an attack, one recording traffic in its proximity and tunneling it to the other node, which replays the traffic. We have incorrect message generation by relay. This attack is often called wormhole attack, see[13].

In a protocol where links are discovered by testing reception, this will result in extraneous link creation (basically, a link between the two "attacking" nodes), see figure 6. This may also break the MPR flooding because of the rule which mandates that a TC message already received from a node which is not an MPR must not be relayed. In figure 7 node B will not relay the TC message from node F since intruder X has artificially relayed this TC message to B.

3.3 Incorrect Traffic Relaying

Nodes in a MANET relay two types of traffic: routing protocol control traffic and data traffic. A node may misbehave by failing to forward either type of traffic correctly.

3.3.1 Incorrect Control Traffic Relaying

If TC messages (or routing protocol control messages in general) are not properly relayed, connectivity loss may result. In networks where no redundancy exists (e.g. in a "strip" network), connectivity loss will certainly occur, while other topologies may provide redundant

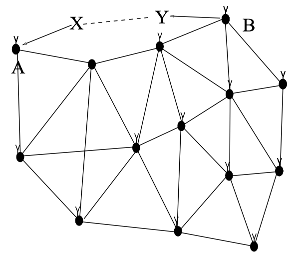

Figure 6: Intruders X and Y are creating an artificial connectivity between nodes A and B

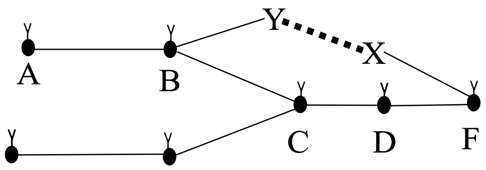

Figure 7: Intruders X and Y are creating an artificial connectivity between nodes F and B, which breaks the MPR flooding in B

connectivity. Similarly if a node does not forward data packets (e.g. if intra-node forwarding is impaired), loss of connectivity may result.

3.3.2 Incorrect Data Traffic Relaying

Even a node correctly generating, processing and forwarding control traffic as required, may act in a malicious way by not forwarding data traffic. The node thereby breaks connectivity in the network (data traffic cannot get through) however this connectivity loss is not detected by the routing protocol (control traffic is correctly relayed).

While this may indeed be due to an attack, this type of situation is also encountered simply due to misconfigured nodes: routing capabilities (through IP forwarding) are typically disabled by default in most operating systems, and must be enabled manually. Failing to do so, effectively, triggers the situation where data traffic is not forwarded/routed while control-traffic (which is forwarded by the action of the routing daemon) is transmitted correctly.

Below is a summary of the identified attacks against the OLSR routing protocol. We have five sorts of attacks :

- incorrect message generation (called in the following IMG attack),
- incorrect message generation with replay (called in the following replay attack),
- incorrect message generation with relay (called in the following relay attack),
- bad data traffic relaying (called BDTR attack),
- bad control traffic relaying (called BCTR attack).

4 Basic mechanisms to secure OLSR

An attack on the ability to provide connectivity in the network must result from the incorrect behavior of, at least, one node in the network. In this context, incorrect means that the node does not process and emit control traffic in accordance with the routing protocol specifications, or that the node does not perform the implied data packet forwarding correctly. We note that in most cases such non-conforming behavior of a node will be due to malice - i.e. specially targeted to interfere with the network connectivity. We call such a node responsible for this incorrect behavior an *intruder*.

In the next section we will need the cryptographic tools which can be used to prevent the intruder from being part of the ad hoc network.

4.1 Cryptographic Requirements

The security architecture proposed is mostly cryptography agnostic. I.e. few constraints are enforced on the cryptographic system employed to secure OLSR as described in this paper. In fact, any cryptographic system, satisfying the following two requirements, may be employed:

- a *signature* for a message can be generated in a node using a function:
  `sign(nodeid, key, message)`
- a *signature* for a message can be verified in a node using a function:
  `verif(originatorid, key, message, signature)`

Public-key as well as symmetric shared-secret key systems can be employed. The properties of various cryptographic systems are beyond the scope of this paper.

We will call *a cryptographic capable node*, a node which has received valid keys and which can sign and verify messages. In an ad hoc network where the nodes meet the above cryptographic requirements, a node is said to cryptographic capable. A cryptographic capable node is said to be compromised if it does not process and emit control traffic in

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 5494 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Sign. Method  |    Reserved   |          MSN Referrer         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                           Time-stamp                           :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                           Signature                           :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 8: OLSR signature message format.

accordance with the routing protocol specifications, or if it does not perform the implied data packet forwarding correctly. Note that a compromised node is, however, a cryptographic capable node.

4.2 OLSR Signatures

To prevent malicious nodes from injecting incorrect information into the network, a signature is generated by the originator of each OLSR control message and transmitted with the control message. In addition, a time-stamp is associated with each signature, in order to estimate a message freshness. Time-stamps are discussed in section 10.

Thus, upon receiving the control message, a node can determine if message originates from a cryptographic capable node, and if message integrity is preserved.

Signatures and time-stamps are genuinely separate entities from OLSR control traffic: while OLSR control traffic serves to acquire and distribute topological information, signatures serve to validate information origins and integrity. Thus, we introduce signatures as a separate type of OLSR control messages, encapsulated and transmitted as described in section 2.2.

Signatures are used by a receiving node to authenticate the corresponding OLSR control message: every control message without a matching corresponding signature is rejected. Depending on the properties of the signature method, different levels of authentication and resilience to attacks can be provided. For instance, the highest level of authentication may be provided by using individual asymmetric keys, as the messages advertised as generated from every non-compromised node are uniquely accepted when they indeed originate from this node. Weaker (but less complex and less computationally intensive) systems can be imagined, e.g. employing a shared secret-key system among cryptographic capable nodes.

In more detail, for each TC or HELLO message generated, a corresponding signature message is generated and transmitted. The format of a SIGNATURE_MESSAGE is specified in figure 8.

To compute a signature corresponding to a TC or HELLO message, the following approach is used. Notice, that this has some similarities with checksum computation:

- the node creates the OLSR control message (HELLO or TC),

- the “current time-stamp” is obtained and updated. Time-stamps are used for replay protection, and are discussed in detail in section 10,
- the signature is computed on the sequence of bytes made up from (i) the TC or HELLO message and (ii) the time-stamp. Notice, that for the computation of the signature, the TTL and Hop-Count fields of the TC or HELLO message are considered as set to 0 (zero) since these fields are modified while the message is in transit and, thus, would otherwise interfere with verification of the message by the receiving node. Thus, the signature is:

```
signature = sign(nodeid,key,
                 < OLSR control message,
                   time-stamp > )
```

Upon receiving a matching message and signature pair, the receiving node verifies the signature thus (again, considering the TTL and Hop-Count of the message to be set to zero):

```
verif(originator address, key,
      < OLSR control message, time-stamp >,
        signature)}
```

If the verification returns true, then the node proceeds to perform time-stamp verification, as described in section 10.

The signature and the time-stamp are contained in an OLSR control message, as illustrated in figure 8 and are transmitted as the data-portion of the general packet format described in section 2.2, with the "Message Type" set to SIGNATURE_MESSAGE, the TTL and Vtime fields set to the values of the TTL and Vtime fields of the message to which the signature is associated.

In order to identify correspondence between a TC or HELLO message, the SIGNATURE_MESSAGE contains in the MSN Referrer field the value of the Message Sequence Number of the control message to which this signature is associated. However, as pointed out in section 2.2, the correspondence achieved by the Message Sequence Number is only unique if possible wraparounds of the 16 bit field are disregarded. This is however not a problem since a further node can use the signature verification to check the correspondence between the control message and the signature message:

- Upon receiving a HELLO or TC message, the node holds the message (for an implementation dependent duration), waiting for the corresponding signature message.
- Upon receiving a signature message, every message held in the previous step, with the same MSN and originator address as the MSN Referrer and originator address in the signature message, is checked for a signature match. If a signature match is found, the time-stamp is further verified, as described in next section. If both signature and time-stamp are validated, the message is accepted and processed following the rules of the OLSR protocol. If not, both the signature message and the control message(s) are held.

The *Sign. Method* fields specifies which method, from a predefined set, is being used to generate the summary of the control message, as well as the actual signature. This includes information about the keys to use, the hashing function used for the signature, and time-stamp methods.

TC messages are forwarded with the associated signature message formed by the originator node of the TC.

To increase security, it can be stipulated that a control message and its corresponding signature message must be in the same packet.

4.2.1 MPR flooding with signatures

Signatures do not protect the TTL and the hop count fields. There is a special reason for that, the computation of TTL and hop count is inherently distributed (i.e. done by successive nodes of the path), so it is not feasible to protect them, unless recording the list of the successive values on the path and signing them.

Since these fields are not protected, they may be attacked: one attack is, for instance, to change the TTL field to 1 before retransmitting the packet, to (possibly) limit its further diffusion to one hop. Note that, as the "hop count" is not used by the OLSR protocol itself (as implemented in RFC 3626), attacking it will have no impact on OLSR.

Attacks on the TTL can be viewed as an instance of general attacks described elsewhere, but several specific solutions can be implemented to prevent such attacks. A few of them are described below:

- Ignore the TTL field entirely. Drawback: a message might travel forever if the network is very large.
- Record the TTL in the duplicate table: if a message has been retransmitted but with a TTL which is noticeably lower than the TTL with which it is received again, it will be transmitted again. Drawback: more transmission and a maximum transmission radius which is lower than initial TTL (around $\frac{initial_{TTL}}{Max_{discrepancy}}$).
- Ignore the TTL, but use the time-stamp to judge if a message is "too old" and should be discarded. Ignoring implementation issues, this is the most logical approach.

4.3 Time-stamps generation

There are numerous ways to generate time-stamps within an ad hoc network. In this section we briefly describe above a very simple way where an authority node distributes a global time. Other more sophisticated approaches are discussed at the end of this research report in a special annex.

The simple time-stamp generation described here uses a centralized approach. Each node periodically receives the current time of an authority node. It then attempts to synchronize its local clock with that time. This time is used toghether with signatures (see next section).

Since the time from the authority node increases monotonically, a standard algorithm can be used for time synchronization:

- each node keeps the last "time" advertised by the authority node;
- when a new "time" is advertised by the signing authority, it is accepted if and only if the new time is strictly greater than the previous ones received. If it is accepted, the clock of the receiving node is updated;
- periodically, each node issues a challenge-response query to the authority node.

This last point, the challenge-response query, is intended to prevent replay attacks with old certificates and old time-stamps in a set of nodes disconnected from the authority node. The challenge-response is classical [28]:

- node A wants to initiate the challenge-response protocol with the authority node;
- node A generates a *nonce*, N_A, a sequence of randomly generated bytes, and sends it in a time query to the authority node. This nonce is signed by node A.
- the signing authority replies with a message containing its current time, and N_A, this time signed by the authority node;
- node A receives the answer, checks for the presence of N_A, and validity of the signature of the authority node. If the signature check and the value of N_A are as expected, node A can now use the net time base.

  If a replay attack was underway, all old replayed messages can now be ignored by A. If node A, after several attempts, receives no answer from the authority node after, it may be suspected that it is disconnected from the signing authority.

The crux is that the attacker which is employing the replay attack is unable to provide a signed message with the "correct" time, correct N_A and a correct signature.

If the authority node is either not sending periodic time-stamps or not responding to challenge response of nodes, a backup authority node can be used. This node can take its turn for instance after the master authority node has not answered its challenge request.

4.4 Mixing signature and time-stamps

Base signatures, which authenticate a message can be complemented with a time-stamp, which holds the time of the node, with sufficient precision. Logically, the time-stamp is appended to the message, then the message and the time are signed, and the time-stamp is distributed along with the signature. Signatures complemented with time-stamps are denoted *full signatures*.

The *full signature* check now comprises two steps:

- the node receiving a message with time-tamp T checks first that $|T - T_{local}| < \delta$, where δ is the maximum time discrepancy accepted, and T_{local} is the time of the local clock.
- if the time-stamp check is passed, the node checks the *base signature*.

Control packets that do not satisfy the two previous conditions are dropped.

4.5 Security overhead

We can mathematically evaluate the overhead increase caused by the sending of signature messages. The size of a HELLO message advertising n nodes varies from $32(n+2)$ to $32(2n+1)$ bits, depending on whether the nodes have the same link/neighbor status or not. The size of a TC message advertising n neighbors is $32(n+1)$ bits.

We assume the use of HMAC-MD5 [16, 26] or DSA [20] for the authentication mechanism, which results in a 128-bit signature for HMAC-MD5 and 320-bit signature for DSA. We also assume the use of a 32-bit time-stamp, which is enough to define the time value for a period of more than 49 days with a granularity of 1 ms. In figure 9 we compare the overhead generated by hello messages with and without signatures (HMAC-MD5 or DSA) with respect to the number of neighbors for a node. Note that we have assumed an average size of $32(1.5n+2)$ for the hello messages. In figure 10, we compare the overhead generated by TC messages with and without signatures (HMAC-MD5 or DSA) with respect to the number of neighbors for a node .

The flowrate of control messages for a node is as follows. With the standard OLSR protocol we have 550 bit/sec, OLSR with HMAC-MD5 signatures results in 752 bit/sec and OLSR with DSA signatures results in 886 bit/sec. These include the computation of IP, UDP and OLSR packet headers (considering IPv4 addresses). For the size of a HELLO, we used the average value. Here we assume that each HELLO/TC is sent alone in a packet for the standard OLSR, and that each HELLO/TC is sent coupled with its SIGNATURE in each packet for the secured OLSR. These figures show that signing control messages introduces a significant extra overhead in ratio. This is not a problem as far as the OLSR overhead remains limited owing to its genuine optimizations.

4.6 What security can be expected from the previous schemes?

Let us assume that we are in an ad hoc network using the above cryptographic tools and that there is no compromised node.

We can be sure that no incorrect traffic generation can be successfully launched. In fact an intruder can not be a cryptographic capable node in the network. Such a node can not be admitted neither as a neighbor nor, fortiori causa, as an MPR of a node. Consequently this node will not be able to interfere with the correct generation of routing tables. For this first kind of attack the use of time-stamps is not mandatory.

For the same reason as above no incorrect message generation with replay attacks can be successful if both the signing and time-stamps schemes are used. The time-stamps scheme

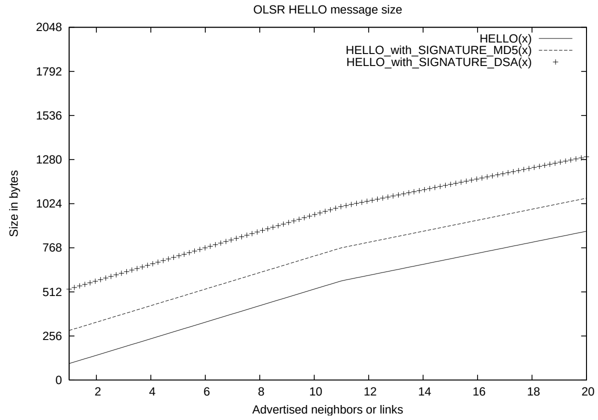

Figure 9: Overhead of Hello messages with and without the signature message

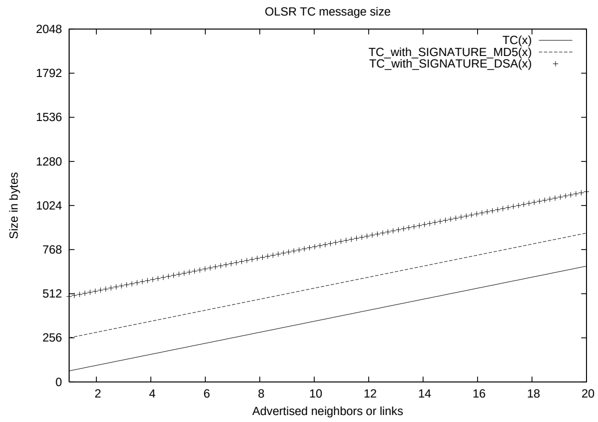

Figure 10: Overhead of TC messages with and without the signature message

will allow cryptographic capable nodes to get rid of control messages sent by intruders replaying old messages.

We are also sure that no intruder can be part of the network. Thus the attacks of bad data traffic relaying or of bad control traffic relaying is not possible since by assumption only an intruder can launch such attacks.

However intruder nodes, although not being cryptographic capable, can relay control packets as described in subsection 3.2.3. Such relay attacks are very difficult to challenge,

attack	IMG	Replay	Relay	BDTR	BCTR
No comp nodes	no	no	yes	no	no
Comp nodes	yes	yes	yes	yes	yes

Table 2: Possible successful attacks with and without compromised nodes

but they are also very difficult to set up. In the appendix we devote a section to studying how the knowledge of node locations can be used to counter such relay attacks.

If we now assume that there is compromised node in the network all the five types of attacks : incorrect traffic generation, incorrect traffic generation with replay, incorrect traffic generation with relay, data traffic relaying, control traffic relaying can be successfully launched. That is the aim of the following section which deals with mechanisms to cope with compromised nodes in an ad hoc network.

Verifying that a node is cryptographic capable implies that one is possession of the keys. How keys can be distributed in ad hoc networks is studied in annex 9.2.3.

5 Coping with compromised nodes

Thus far, we have been exclusively concerned with keeping intruders out of the network, i.e. stopping control traffic from intruder nodes from being diffused among other nodes. The prime hypothesis has been that a node which is not an intruder is known to behave correctly. Specifically, we have assumed that control traffic is generated, processed and forwarded in accordance with the protocol specification, and that data traffic is forwarded correctly. This is illustrated as the topmost situation in figure 11.

In this section, we change that hypothesis in that we will investigate how a network can be protected against two different problems: a node which has been compromised, or/and imaginary links were created with the relay attack 3.2.3 (these are *compromised links*). If we are in an architecture as described in annex 9.2.3, the first case means that we have a node whose key and identity is known by the signing authority, but which for some reason has stopped behaving correctly (e.g. sending or forwarding control traffic incorrectly, forwarding data traffic incorrectly etc). This is illustrated as the middle situation in figure 11.

Handling the situation where a compromised node and/or some compromised links are present in the network implies two separate issues, namely *detecting* that a node or link is misbehaving, followed by a *corrective action*, allowing the non-compromised part of the network to continue operating correctly.

The proposed corrective action for a compromised node is illustrated in the bottommost situation of figure 11. The compromised node is marked as "excluded" and is thus excluded from the network. Similarly, the corrective action for compromised links is to exclude them.

In this section, we will first discuss how to detect a compromised node or a compromised link in the network. Then, we will address issues regarding the proposed corrective actions.

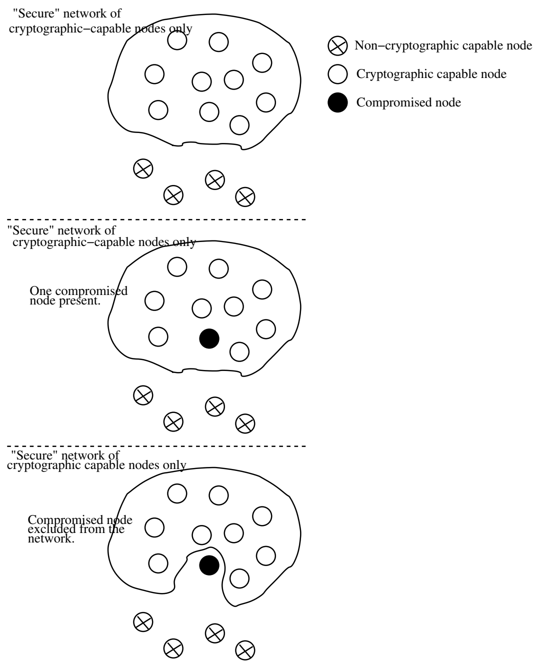

Figure 11: Top: the network is excluding all non-"cryptographic capable" nodes, assuming that "cryptographic capable" nodes are well behaved. Middle: a "cryptographic capable node" has become compromised and thus misbehaves. Bottom: upon *detection* of the compromised node, a *corrective action* renders the compromised node no longer recognized as "cryptographic capable" and the mechanisms described in section 5.4 exclude the compromised node from the network.

5.1 Detecting Link Spoofing in HELLO- and TC-messages

In section 4, we described basic mechanisms which aim to keep intruders outside the network when there are no compromised nodes in the network. However with compromised nodes present, intruders may be in possession of given keys of the networks and thus they can issue valid signatures for non-existing links. To prevent this situation, a more fine-grained signature mechanism is required. This mechanism has been already presented in [24].

While HELLO and TC messages, in terms of the routing protocol functioning, have both a different structure, scope and purpose, they are identical in terms of preventing link spoofing: both advertise links between the originator of the message and a set of neighbors to the originator. Hence, the same general considerations apply for both message types. In the following discussion, we will therefore use the term "OLSR control message" meaning either a HELLO or a TC message.

In general, OLSR control messages have the semantics of the originator advertising "I have a link with these other neighbor nodes". The signature on the message, introduced in section 4, serves to verify that the originator is indeed the one claiming such a link to exist. The task in hand is to ensure that it can be validated that the neighbor also believes that such a link exists.

Considering OLSR, we observe that:

- MPRs are selected from among the nodes with which there exists a symmetric link;
- TC messages contain symmetric links only;
- MPR selection is accepted from nodes with which there exists a symmetric link.

Hence, only nodes with which a symmetric link exists can affect the network formation and functioning. Furthermore, if an asymmetric link exists between two nodes, i.e. node A can hear node B, then the link is, indeed, at best asymmetric – and thus only node A can make any statements regarding the link.

We therefore propose a simple mechanism which, in collaboration with OLSR neighbor sensing, will allow not only Bi-directionality-checks, but also the ability to advertise "verifiable symmetric links". By a "verifiable symmetric link" we mean a link, which the nodes in both ends have signed as being valid for a given amount of time. Our approach is illustrated in figure 12.

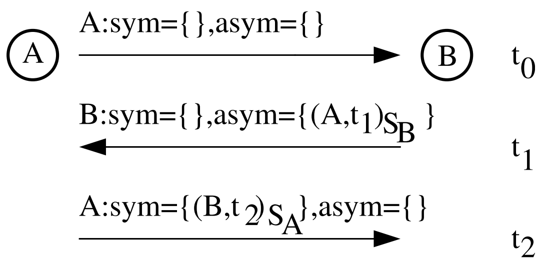

Figure 12: Bidirectionality check with signatures.

At network initialization, t_0, node A sends a HELLO message. This is received by node B. Node B, in its next HELLO message, indicates that it has heard node A (status: asymmetric). It furthermore attaches its time-stamp, signs the pair (A, t_1) and transmits $(A, t_1)_{S_B}$. Upon receiving this information, node A now has node B's signature as proof that at time t_1 there existed a link from node A to node B.

When node A at some point wishes to advertise this link in an OLSR control message in a way such that it can be verified by other nodes in the network, it will advertise $((A, t_1)_{S_B})_{S_A}$. This indicates to the recipient that at the time t_1, nodes A and B both claim to be able to receive messages from each other. The double signature provides authentication by both sides.

Hence, upon receiving an OLSR control message containing such signed links, the recipient can, based on the signatures and the time-stamps, ensure that the originator of the message is transmitting valid and correct, and decide to process or discard the message accordingly. If a node is found to send "incorrect" information – e.g. to spoof links to another node – an inconsistency is reported to the security authority, triggering a corrective action to degrade the offending node to "excluded" and remove it from the network.

With at most one compromised node, the net result is that no non-existing link can be injected in the network. Note that this excludes compromised links created by relay attack (section 3.2.3), which exist to the extent that they are created by genuinely relaying packets back and forth. In general, the mechanism presented here, of advertising verifiable links, ensures that the only nonexisting links that could be injected into the network would be between two compromised nodes.

5.2 Detecting Incorrect Data Traffic Relaying

The previous section details how to prevent and detect attacks on information at the routing protocol level. In this section, the focus is on detecting and alleviating incorrect forwarding of data packets. Because incorrect routing information is detected, and prevented by the provisions specified in the previous section, the central hypothesis for this section is that the routing information is correct (the only exception being for compromised links created by relay attack 3.2.3).

Two cases are represented in figure 13.

In case 1, there is a relay attack. An imaginary link is created by node C, which is invisible for nodes A and B. Node C may drop packets, change their order, replay them, or inject miscellaneous packets. This is a *compromised link*.

In case 2, there is a *compromised node* C. This node might perform the same actions as the attacker node C in case 1. The situation is, however, different because on the one hand, the node is directly visible, but on the other hand, detecting misbehavior requires gathering information from nodes which are two-hops apart.

In case 2, it doesn't matter if one or both of the links are actually also compromised links. Any misbehavior of the link can be translated to an equivalent to a misbehavior of the compromised node.

5.2.1 Detection through Network Flow Conservation

BDTR and BCTR attacks can be countered in different ways. One example is to use overhearing of transmissions to detect incorrect forwarding behavior, using the watchdog/pathrater approach of [18]. Other approaches use detection through acknowledgments or probe packets.

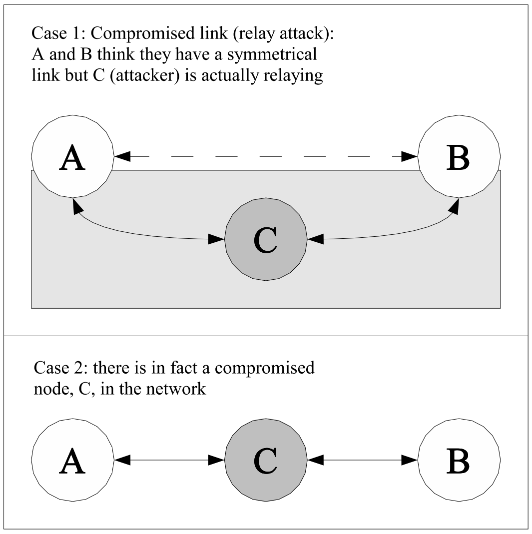

Figure 13: Forwarding attacks

Another line of approach is based on *network flow conservation* [5] and [2]. It is not sufficient to provide resilience to attacks, as the closer analysis of [14] shows, but it provides some minimal detection.

The law of "Conservation of Flow" states that: *an input must either be absorbed or sent on as an output* [14]. This is the basis for a set of algorithms which essentially count packets received and sent by one node to each of its neighbors. Then exchanging, or collecting these counts makes it possible to check whether the node is exhibiting proper routing behavior. Essentially, 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.

5.2.2 Applying Network Flow Conservation

In our situation, we are concerned only with the two cases illustrated in figure 13.

In case 1, the law of conservation of flow translates into:

- Number of packets sent by A to B = number of packets received by B from A. This is flow conservation applied to a link.

Case 2 is more complicated. Let's assume the send/receive packet counts are consistent considering each pair involving C and its neighbors (i.e. ensuring that case 1 checks are passed successfully). Now, nodes A and B should compute statistics on the *transit packets of* C. A transit packet of node C is a packet that is neither destined for C nor originating from C (as in [2]).

Assuming the following data is recorded:

- For each node N_i, the number of transit packets sent to C
- For each node N_i, the number of transit packets sent from C

The law of conservation (on a node) roughly translates as : the sum of the number of transit packets sent to C by all other nodes N_i, is equal to the sum of the number of transit packets sent by C to all other nodes N_j. It ensures that node C has emitted as many transit packets as it has received.

5.2.3 Precise Network Flow Conservation Check

To be more specific, and again following [2], the law of conservation should be written more precisely, using more counters, to account for the non-transit traffic. Each pair of neighbors (X,Y) records the following information about the packets which went/should have gone from X to Y:

- The number of packets which are transit packets for both X and for Y (i.e. neither source nor destination are X nor Y): $T_{X\to Y}$.
- The number of packets with source X which are transit packets for Y: $S_{X\to Y}$
- The number of packets with destination Y which are transit packets for X: $D_{X\to Y}$.
- The number of packets which were misrouted by X to Y: $M_{X\to Y}$. Misrouted packets are packets forwarded by X to Y although Y is no closer to the destination than X.
- Additionally, the total number of all packets without regard for the source or destination: $A_{X\to Y}$

Each quantity can be seen either from node X's perspective or from node Y's perspective: we denote $T_{X\to Y}$ for instance, the following way: $T_{X\to Y}[X]$ for X's perspective, and $T_{X\to Y}[Y]$ for Y's perspective. Similarly, for instance, $M_{X\to Y}[X]$ would be the number of packets misrouted by X to Y from the perspective of node X, and incidentally should thus normally be 0.

The complete flow conservation equation for a node A is obtained writing the expression of `output - input`, which is no longer 0 here and is actually equal to
`produced packets - consumed packets`

Thus we have:

```
packets sent by A to neighbors
- packets sent to A by neighbors
= packets sent by A originating from A
- packets sent to A destined for A
- packets sent to A that A judged misrouted
```

This translated into the following equation for a node A, and the set of its neighbors N_i:

$$(\sum_i A_{A\to N_i}[N_i]) - (\sum_i A_{N_i\to A}[N_i])$$
$$= (\sum_i S_{A\to N_i}[N_i]) - (\sum_i D_{N_i\to A}[N_i]) - (\sum_i M_{N_i\to A}[N_i])$$

5.2.4 General case: weaknesses of network flow conservation

Flow conservation protects from some attacks, namely traffic relaying attacks which simply drop packets. Still, [14] shows a number of attacks that are not detectable using this technique. The central weakness is that assuming the *proper number of packets* was forwarded by a node, does not prove that the *proper packets* were sent.

Detecting such attacks requires data authentication and, additionally, identifying which packets were received and sent by a node or link. An expensive possible solution would be, for example, to keep a list of signatures of packets received and sent, instead of a count.

5.2.5 Specific case: strengths of network flow conservation

In the more specific case where none of the nodes is compromised, network flow conservation still makes it possible to perform simple detection of the remaining possible attack, the relay attack (i.e. "compromised links").

The relay attack would create links that do not exist or where data or control traffic is not transmitted consistently. This is easily detected by the simpler "equation" of network flow conservation:

```
number of packets sent by A to a neighbor B = number of packets sent
by B to a neighbor A.
```

Such information can be exchanged with the equivalent of HELLO messages complemented with packet or message counters. In order to address the problem of the proper *number* of packets being relayed by a link, but not the proper packets, there are several solutions. It is possible, for instance, to use approaches such as Bloom filters, to represent a (hashed) set of signatures of sent/received packets ; or to perform authentication on packets (and not only OLSR messages) with replay protection.

5.3 Detecting Incorrect Control Traffic Relaying

In MPR flooding, the responsibility of diffusing the messages to the entire network is distributed. As a consequence, misbehavior is harder to detect and, if detected, to attribute to one single node.

On the scale of individual transmissions, the compromised links dropping packets during MPR flooding are detected with the same techniques as described for data traffic, using counts at the link level. To accommodate broadcast/flooding traffic, each node must additionally report how many broadcast/flood messages it has sent and received from each neighbor. The number of broadcast messages received by a node from a given neighbor, should be equal to the number of of broadcast messages this neighbor has sent to the network.

On the scale of the entire network, compromised nodes dropping packets during MPR flooding, or performing incorrect MPR selection, are harder to detect. A remedy to this problem is the *MPR_COVERAGE* option of OLSR, allowing each node to cover its two hop neighborhood by at least two MPRs: at most one of them will be compromised, and the non-compromised node will ensure two-hop coverage. This is at the expense of MPR flooding efficiency and under the assumption of a single compromised node present in the network.

Another subtle attack on flooding exists with respect to compromised links. This attack would attach itself to the "first transmit rule" of MPR flooding[1].

By manufacturing an inconsistent reception order of packets of the network, a set of cooperating compromised links might block MPR flooding, by having nodes stopping retransmission due to that rule. A quite expensive solution would be to remove the first transmit rule.

A final misbehavior in message forwarding exists, although they are preventable: attacks on the mutable fields of MPR messages (the "time-to-live" field). A compromised node forwarding a message could modify the TTL field to an artificially low value, causing the message to be dropped prior to reaching the entire network. A possible solution is to simply ignore this field. The time-stamp, present in the signature of each message ensures that a message can be checked for freshness and, if too old, be dropped/rejected.

5.4 Corrective Action

In this part we describe an optional architecture, which can use the previous different detections of node misbehavior, to carry out corrective action. It is based upon a centralized Security Authority. The Security Authority is assigned the task of collecting information, detecting misbehaviors, arbitrating them and issuing actions to remedy that misbehavior.

The information collected, and the mechanisms for detecting misbehaviors are specified in their relative parts in sections 5.1 and 5.2. A general procedure to report information, in the possible presence of one compromised node, is nevertheless specified in the next section.

Once the security authority has collected the information from the nodes, and detects (or acknowledges) an inconsistency, which it interprets as a misbehavior, it may take several decisions: a limited action, suppressing links, when two nodes' information are in disagreement, a definitive action, when a node is identified as misbehaving. A section is devoted to the presentation of each action.

[1]The "first transmit rule" states, that if a message has been received by a node for which it was not selected as MPR, it will never be retransmitted by the node even if later received from an MPR selector – see [8] for further details

Information	Reporting mechanism	Possible associated attack
Traffic counters for flow conservation	Periodic transmission (acknowledged)	Incorrect traffic forwarding
Incorrect signature of one message or packet	Alarm with pure flooding	Forgery attempt or use of revocated key
Incorrect signature of a verifiable link	Alarm with pure flooding	Link spoofing

Table 3: Information reported by nodes to the Security Authority

5.4.1 Reporting Information to the Security Authority

When a node reports information to the security authority which will be used as a basis for detecting misbehavior, the central problem is that a compromised node might also prevent that information from reaching the security authority in the first place. The classical solution to such a problem is to use pure flooding to report a problem. This will ensure that the information will get through if there exists an uncompromised path from the reporting node to the security authority.

However if the reported information is frequent, and does not immediately lead to the conclusion of the existence of a misbehavior, pure flooding might be too expensive. In this case, an acknowledgment mechanism can be used. Initially, the node reports information to the security authority using normal transmission (unicast transmission in data packets) ; the security authority acknowledges the information packets it has received from each node, by listing their signatures. If the node detects that its information has not been acknowledged, it falls back to pure flooding. This scheme is particularly suited to periodic transmission of information from nodes, as the central authority can piggyback the acknowledgments on the certificates/base time periodic diffusions.

Turning now to the details of the nature of the information itself, which is reported by the nodes to the security authority, table 5.4.1 summarizes the different types of information used:

5.4.2 Arbitration of Flow Conservation Violation

Section 5.2 highlighted the principles underlying arbitration of flow conservation, and detailed which counters were gathered by the security authority.

The counters are updated on time intervals whose start and end are set by the security authority. This is greatly eased by the fact that the signing authority is also the source for time synchronization. Once gathered, the security authority can perform the check in three steps.

Step 1 Checking for misrouted packets: For each pair of nodes X, Y once the number of packets from Y misrouted to X (according to X) is greater than a certain threshold, the link is considered to be compromised.

Step 2 Checking for case 1 (conservation for links): For each pair of nodes X, Y which transmitted traffic to each other, same counter values are known, both from the perspective of X and from the perspective of Y. If they differ by too wide a margin, the link is considered to be compromised - action is initiated to remove it (see section 5.4.4). In reality either it is "compromised" or one of the nodes is.

Step 3 Checking for case 2 (conservation for nodes): For each node in the network, which had no link removed in the previous steps, the flow conservation equation of section 5.2.3 is checked. The two sides of the equation should be equal ; again, if the discrepancy is greater than a certain constant, the node is strongly suspected to be compromised.

Note that in the checking, in step 2, since at most one node is compromised, at least one side of the link is not compromised and sending correct results. As a consequence, even if the other side of the link is compromised, it cannot send (too) incorrect counter reports, for it would be detected. This ensures that step 2, uses information that is not too far from reality, and thus prevents false accusations.

5.4.3 Arbitration of Alarms

The security authority receives a number of alarms, which leads to different degrees of doubt about a given node or link. The processing of the previous section highlights discrepancies which are some other sources of alarms. The sources of alarms are summarized in table 5.4.3.

In some cases, the dynamic nature of the network should be considered - for instance mobility can contribute to packet drops, and results are not definite. In some cases, a misbehavior is established but the source of the misbehavior is not known with certainty. Thus in many cases, the security authority must perform arbitration.

Several solutions related to similar problems have been developed in recent articles. In the CONFIDANT [3] protocol, when a node becomes aware of an offending node it may issue an alarm to inform a centralized entity. Alternatively, in the Watchdog-Pathrater scheme [19], each node privately keeps a rating for each of the other network nodes in the network. Nodes' ratings are calculated during route discovery. In both cases, once the offending nodes are detected, one tries to find path that possibly avoids offending nodes.

Part of the problem lies in the difficulty of evaluating node A's statements about node B, how can one be sure that node A is not lying? A related critical problem when exchanging "trust" evaluation between nodes is how to distinguish false alarms from correct ones as a

Alarm	Consequence
Incorrect signature of one message or packet	high doubts about node(s) but requires further arbitration to identify guilty node
Incorrect signature of a verifiable link	high doubts about node, and reporting packet is enough to establish attack
Reported misrouted packets	doubts about nodes, requires arbitration
Inconsistent link flow conservation	high doubts about link, some doubt about nodes
Inconsistent node flow conservation	high doubts about the link

Table 4: Alarms and associated severity

compromised node can also issue an alarm. In the CONFIDANT protocol, the problem is lessened by timeout and subsequent recovery of nodes that have behaved well for a specified period of time.

To implement this effect, the security authority maintains a trust rating for each of the other network nodes and links, based on alarms it has received. Each time a valid alarm is received the trust rating is decreased. Under some established algorithm the security node can take decisions for corrective action: removing links or removing nodes (either temporarily or permanently). These two actions are detailed in the following two sections.

5.4.4 Limited action: removing links

The security authority does so by periodically transmitting a blacklist of links to the entire network. All nodes should ignore such links, in the processing of OLSR.

5.4.5 Definitive action: removing nodes

When the security authority pursues the definitive action of excluding a node from the network, this is done by revocating the key of the node:

- for some duration, the security authority explicitly advertises, along the periodic certificates diffusion, the key which has been revocated.

- later, once the lifetime of the revocated key has expired, the security authority no longer advertises it.

This mechanism is similar in spirit to the `LOST_LINK` advertisement mechanism in OLSR which advertises the disappearance of a link.

6 Using node locations to counter relay attacks

If we assume that each node in the ad hoc network knows its location, this information can be sent together with the control messages. Knowing this information when a control message is received can be used to discard suspicious control packets e.g. control messages coming from a too distant a node. This section is based on this idea and uses techniques already published in [25].

The node position can be obtained by a positioning system device (e.g. GPS) embedded in the node hardware. A positioning system usually also provides time synchronization; see [17]. Other solutions where the nodes are not all equipped with a positioning system [27] or even where no positioning system is used at all [4] can be envisioned. However, due to the possible presence of malicious nodes, solutions which rely on feedback or signals from other nodes (e.g. the emission power) cannot be considered as safe.

If the nodes can use directional antennae instead of omni [9] a greater control concerning the likeliness of received control messages can be implemented. This will be shown in the following.

6.1 Specifications

We suggest therefore some modifications to the basic mechanisms described in 4. A SIGLOC (which stands for SIGnature and LOCalization) control message substitutes the SIGNATURE_MESSAGE; the former includes a new field "node location", which contains the current geographical position of the sending node as obtained from a positioning system e.g. GPS. This field is 32 bits long (which is enough to define the position over an area of more than 4200 square km with a granularity of 1 m), and is included in the signature computation. The message format is given in figure 14. This mechanism requires the deployment of a Public Key Infrastructure and a time-stamp synchronization algorithm between all nodes. These topics are discussed further in this paper; please refer to Annex I and II for more details.

A node informs the other nodes about its current position in a SIGLOC message (which, we recall, is sent with every generated HELLO and TC). The receiving node first couples the SIGLOC with its companion HELLO/TC and verifies the correctness of the time-stamp and signature, as specified by the protocol in [1]; then it extracts the position information and stores the tuple $\langle$ node address, position, time-stamp $\rangle$ in a *position table*. For each node, the most recent position is memorized in the position table. The position of the originator node and the originator of a TC message may not be within reach. Thus a subtle relay attack can be launched against OLSR because OLSR mandates that a TC message is not relayed when it has already been received from a non MPR selector node. Thus the MPR flooding can be artificially broken by this attack. A way to repel this relay attack would be to also sign the OLSR packet, hop by hop.

The advantage of knowing the geographical position of nodes is that a receiver node can speculate whether a link is likely or not. This link may be a direct link with a neighbor or a link advertised in a TC message.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 5494 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Sign. Method  |    Reserved   |         MSN Referrer          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          node location                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          timestamp                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                          Signature                            :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 14: SIGLOC message format.

6.1.1 Checking the originator of Hello messages

Let p_A be the current position of the receiver A, and t_A the current time according to its clock when A receives a control message from node B. Node A learns from the SIGLOC message the position of node B at time t_B. Let us call Δt the discrepancy in the network nodes clocks' synchronization, Δd the maximum absolute error in position information, v the maximum velocity of any node in the network, and r the maximum transmission range. Taking into account errors and nodes motion $\|p_A - p_B\|$ must satisfy the following inequality:

$$\|p_A - p_B\| \leq r + (t_A - t_B + \Delta t) \cdot 2v + 2\Delta d = r + 2i \tag{6.1}$$

When equation 6.1 is not valid, it means that the receiver node must be too far from the sender node to be able to hear its transmission. Therefore such a Hello message is highly suspicious and might well be tunneled in a relay attack. The receiver should drop such a Hello message.

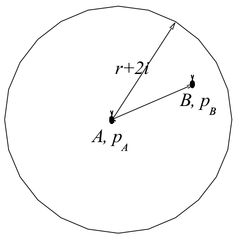

Figure 15: Test of likelihood when a hello message is received

When directional antennae are used, the receiver node can know which direction the signal is coming from. Based on $p_A - p_B$ and using simple geometry, this allows the receiver node to estimate the correctness of the position p_B declared by the sender node. If one denotes by $\overline{AB}$ the vector linking node A and node B when the hello message is received, we should have:

$$\|p_B - p_A - \overline{AB}\| \leq \ (t_A - t_B + \Delta t) \cdot 2v + 2\Delta d = 2i. \tag{6.2}$$

This information can be useful if the transmission range r is not known precisely. In such a case it is possible to use a lower bound r_{lb} on the transmission range and derive from equation 6.2 the sector in which the sender should be. Should the directional antennae indicate another direction for the reception, such a transmission should be considered as a fake. The receiver should drop such a Hello message.

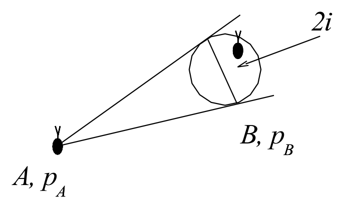

Figure 16: Test of likelihood when a hello message is received using a directional antenna

The previous inequalities can be refined if we have a better estimation of a node's velocity. If we assume that a node's velocity is linked to the validity time given in the OLSR packet, we can refine equation 6.1 or equation 6.2 using a better estimation of the velocity of the nodes A and B.

6.1.2 Checking links advertised in Hello and TC messages

Equation 6.1 also makes it possible to detect and reject the incorrect control messages described in section 3.2. Let us now assume that *node C is receiving a control message sent by node A advertising a link between node A and node B*, see figure 17. In equation 6.1, p_A is the location of the originator node, node A. The position of node A is found in the SIGLOC message to be p_A at time t_A. p_B must be the location of node B at time t_B. The location of node B can be found in node C's location table. Such a location should be found for a given time t_B which minimizes $|t_A - t_B|$. An interpolation for the position of B can be used if the location of node B is not known for a time close to t_A.

If equation 6.1 is not satisfied then the link $A - B$ must not be registered in the neighbor or topology table and the control message advertising this link should be dropped.

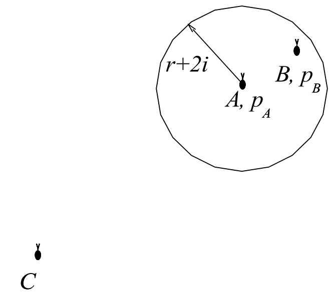

Figure 17: Test of likelihood for links advertised in Hello or TC messages

6.2 Detailed procedure to sign messages

When node A generates a HELLO or a TC, it must also generate a SIGLOC and performs the following steps:

1. create the SIGLOC message
2. insert the node's position
3. insert the time-stamp corresponding to the actual time
4. compute the signature on the HELLO/TC message and write it in the SIGLOC message
5. send the HELLO/TC and the SIGLOC

6.3 Detailed procedure when receiving a control message

When a node receives a control message (Hello or TC) originating from A, the following actions should be carried out:

1. correctly pair off the HELLO/TC with its SIGLOC companion, by matching the Message Sequence Number with the MSN Referrer
2. check the freshness of the time-stamp
3. check the validity of the Signature with the convenient key
4. check the validity of a hello message with respect to its originator node according to 6.1.1 and the validity of links advertised by a hello message or a TC message according to 6.1.2.
5. store the tuple $\langle$ address of A, node location, time-stamp $\rangle$ in the position table

6.4 Protection offered

A simple signature with time-stamps is sufficient to avoid the consequences of attacks of incorrect traffic generation and of incorrect traffic relaying under the assumption that intruder nodes can not sign control packets. Using nodes' location in the signature will make it possible to counter relay attacks.

6.4.1 Repelling relay attacks

Let us analyze the consequences that may be drawn from equation 6.1. If we use figures such as $v = 60km/h$, $t_A - t_B + \Delta t \leq 100ms$ and $\Delta d = 1m$ we find that the originator node must be within a radius of $r + 5.333m$. When r is not too small (e.g $r > 50m$, the control packet received is necessarily sent by a node nearly within the coverage of the recipient. That means that relay attacks using such a control packet will be on the one hand difficult to launch because the real control packet is likely to be heard by the recipient and, on the other hand, not very efficient since the node whose control message is relayed, will probably be, at most, two hops away. When r is small (e.g $20m < r < 50m$) the use of information given by a unidirectional antenna can be useful since the sector in which the signal is expected remains of a reasonable size.

It can noticed that a relay attack where a TC message is tunneled from one point of the network to another can not be repelled since the signature holds the position of the originator node and the originator of a TC message may not be within reach. Thus a subtle relay attack can launched against OLSR because OLSR mandates that a TC message is not relayed when it has already been received from a non MPR selector node. Thus the MPR flooding can be artificially broken by this attack. One way to repel this relay attack would be to also sign the OLSR packet, hop by hop.

6.4.2 Repelling incorrect traffic generation

The equation 6.1 also makes it possible to reject control messages advertising impossible links because the two endpoints are too far from each other. Thus the SIGLOC signature can repel attacks of link spoofing. However under the assumption that only usual nodes can conveniently sign control packets, attacks of incorrect traffic generation should have no effect of the integrity of the network. The SIGLOC signature should nonetheless be useful under the assumption of compromised nodes. With such an assumption, control messages advertising impossible links can be discovered.

7 Conclusion

We have identified for OLSR five attacks on the network's integrity: the incorrect control message generation attack, the replay attack, the relay attack, the bad data traffic relaying and the bad control traffic relaying attacks All these attacks may have important consequences on network connectivity.

Under the assumption that there is compromised node in the network, the signature and time-stamps mechanisms described in section 4 counter all the identified attacks except the relay attack. The nodes'knowledge of their own position can be used to mitigate this latter attack.

If we assume that there are compromised nodes in the network, securing OLSR is much more complex. However the general idea is still to detect compromised nodes or links and to remove such nodes or links. Perfect securisation under such an assumption seems out of reach. However using only verifiable symmetric links and checking flows conservation can solve numerous configurations of attacks.

References

[1] Cedric Adjih, Thomas Clausen, Philippe Jacquet, Anis Laouiti, Paul Mühlethaler, and Daniele Raffo. Securing the OLSR protocol. In *Proceedings of Med-Hoc-Net*, Mahdia, Tunisia, June 25–27 2003.

[2] K. A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R. A. Olsson. Detecting disruptive routers: A distributed network monitoring approach. pages 115–124, 1997.

[3] S. Buchegger and J. Le Boudec. Performance analysis of the confidant protocol: Cooperation of nodes — fairness in dynamic ad-hoc networks, 2002.

[4] Srdan Capkun, Maher Hamdi, and Jean-Pierre Hubaux. GPS-free positioning in mobile ad hoc networks. In *HICSS*, 2001.

[5] Steven Cheung and Karl Levitt. Protecting routing infrastructures from denial of service using cooperative intrusion detection. In *New Security Paradigms Workshop*, 1997.

[6] Thomas Clausen, Gitte Hansen, Lars Christensen, and Gerd Behrmann. The optimized link state routing protocol - evaluation through experiments and simulation. In *Proceeding of Wireless Personal Multimedia Communications*. MindPass Center for Distributed Systems, Aalborg University, Fourth International Symposium on Wireless Personal Multimedia Communications, September 2001.

[7] Thomas Clausen, Philippe Jacquet, and Laurent Viennot. Investigating the impact of parital topology in proactive manet routing protocols. In *Proceeding of Wireless Personal Multimedia Communications*. MindPass Center for Distributed Systems, Aalborg University and Project Hipercom, INRIA Rocquencourt, Fifth International Symposium on Wireless Personal Multimedia Communications, November 2002.

[8] Thomas Clausen (ed) and Philippe Jacquet. RFC 3626: Optimized link-state routing protocol. Internet Engineering Task Force, Request For Comments (experimental), October 2003.

[9] Rob Flickenger. *Building Wireless Community Networks*. O'Reilly & Associates Inc., 2003.

[10] Manel Guerrero Zapata and N. Asokan. Securing Ad hoc Routing Protocols. In *Proceedings of the 2002 ACM Workshop on Wireless Security (WiSe 2002)*, pages 1–10, September 2002.

[11] http://www.cryptopp.com.

[12] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking*. Rice University, MobiCom 2002, September 2002.

[13] 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.

[14] John R. Hughes, Tuomas Aura, and Matt Bishop. Using conservation of flow as a security mechanism in network protocols. In *IEEE Symposium on Security and Privacy*, pages 131–132, 2000.

[15] J. G. Jetcheva, D. Johnson, D. Maltz, and Y.C. Hu. Dynamic source routing (DSR). Internet Draft, draft-ietf-manet-dsr-08.txt, February 24 2003, Work in progress.

[16] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-hashing for message authentication, February 1997. RFC 2104, Informational.

[17] Trimble Navigation Limited. Data sheet and specifications for thunderbolt GPS disciplined clock, 2000. `http://www.trimble.com`.

[18] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *In Proceedings of the Sixth annual ACM/IEEE International Conference on Mobile Computing and Networking pages 255–265*, 2000.

[19] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[20] NIST. Dss digital signature standard. Processing Standards Publication 186-2, January 2000.

[21] R. Ogier, M. Lewis, and F. Templin. Topology dissemination based on reverse-path forwarding (TBRPF). Internet Draft, draft-ietf-manet-dsr-07.txt, March 3 2003, Work in progress.

[22] C. E. Perkins, E. M. Royer, and S. R. Das. RFC 3561: Ad hoc on-demand distance vector (AODV) routing. Internet Engineering Task Force, Request For Comments (experimental), July 2003, Work in progress.

[23] A. Qayyum. Analysis and evaluation of channel access schemes and routing protocols in wireless lans. PhD thesis, University of Paris-Sud, Orsay, France, 2000.

[24] 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.

[25] 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.

[26] R. Rivest. The MD5 message-digest algorithm, April 1992. RFC 1321.

[27] 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. ACM Press, 2001.

[28] Bruce Schneier. Applied cryptograpy, protocols, algorithms and source code in c. John Wiley and Sons, 1994.

8 Annex I: Time-stamp generation

8.1 Time-stamps with precise clocks

When a precise clock is available (such as the clock obtained by means of GPS information, or by atomic clocks), a simple choice for the time-stamp is to use real clock time.

Then, as described in section 4.4, a node receiving a message with time-stamp T checks that $|T - T_{local}| < \delta$, where δ is the maximum time discrepancy accepted, and T_{local} is the time of the local clock.

8.2 Time-stamps with imprecise clock or with counters

Another way to generate a time-stamp is to use imprecise clock information or counters. By imprecise clock information, we mean time information with a clock which may drift, and hence which would require (periodic) synchronization to be kept in sync with real time.

Several options are possible, for instance:

- Manual setting of clock information. While having the disadvantage of not being automatic obviously, as the drift may be much lower than the tolerance margin for time discrepancy for messages, this may be a practical option, possibly combined with alarms.
- Using a distributed counter. A time-stamp "counter" is kept: it is incremented each time a packet is sent. Because all nodes must agree on the counter, some kind of synchronization must be provided. For instance taking the "maximal" observed value of the time-stamp counter. This is only possible under the assumption that no node is compromised.

Because imprecise clocks or counters are use information that is updated in a distributed way, there is a general need to synchronize that information. However, this is a chicken and egg problem at network initialization (or when there is a network merge), as synchronization or election between all (or the majority) of nodes in the network would require authentication, which itself would require time-stamps (to prevent replay attack).

The root of the problem is that, in this context, the node would require a proof of freshness before trusting any information it receives: this is normally done by sending some data in the network (like the nonces), and checking afterwise traces of that data in further received information. In a distributed system, this is complex, and can achieved by the system given in the next section.

8.3 Distributed methods for time-stamp exchange

In this section, a distributed method for time-stamp exchange is given.

We assume that an asymmetric key system is used. However, no other specific hypotheses are assumed and as a consequence:

- Any number of nodes might be compromised (from the point of view of the time-stamp algorithm).
- Replays attacks might be launched.

The basic algorithm relies on three principles:

- Each node has its own time-stamp. The intent of this is to limit the impact of compromised nodes.
- The time-stamps are exchanged with a variation of the basic challenge-response protocol (*handshake*).
- Once exchanged, the time-stamps are periodically broadcast to the network.

There are possible variations in the mechanism used, and more details are given in the following sections.

8.4 Time-stamp

The time-stamps themselves of each node may be counters or (drifting) clocks.

There are actually two kinds of time-stamps: time-stamps are monotonically increasing and do not wrap around ; and others. The first kind are more difficult to implement, especially with respect to initialization, but offer more optimization potential other the second ones. In the following, we denote the first ones, as *"strictly increasing time-stamps"* and the others as *"time-stamps with wrap-around"*.

The implication of each node having a different time-stamp, is that each node holds a list of time-stamps of all the other nodes.

8.5 Distributed time-stamp protocol

8.5.1 Time-stamp exchange: handshake

The time-stamp exchange between two nodes has really two stages in itself: in the first one, the nodes do not have proper information about each other's time-stamps and hence should exchange it. It is called an *handshake*. In the second stage, they have, and they are able to send updated time-stamp information (equivalent of "clock drift correction" in wall clock terms) using the time-stamp information they already have.

The handshake is described in this section and the second stage is described in the next section.

The handshake consists of a two-way exchange of time-stamp between two nodes A and B, which is formally [2]

1. $A \to B : \{T_A\}_{S_A}$

[2] Note: the messages may need to also include identifiers of A and B in exchanged messages inside the signed part, to prevent some attacks

2. $B \rightarrow A : \{T_B, (A, T_A)\}_{S_B}$; and then the handshake is completed from A's point of view.

where T_A is the time-stamp of A and $\{...\}_{S_A}$ is a message signed by the public key of A.

To complete this handshake from B's point of view, A will send the message:

1. $A \rightarrow B : \{T_A, (B, T_B)\}_{S_A}$

Under the assumption that any node can be compromised, two nodes A and B need to communicate directly with each other (with pure flooding if they are several hops away), to perform this handshake.

8.5.2 Time-stamp exchange: update

If we now assume that nodes have performed this handshake, then they can adjust their time-stamps with periodic broadcast of all the tables of all the time-stamps they know:

1. Time-stamp update message: $A \rightarrow all : \{T_A, (B, T_B)_{S_B}, (C, T_C)_{S_C}, \ldots\}_{S_A}$

That way, a node, for instance C, receiving this message, will:

- first, see that the message is a valid message from A from the time-stamp of A according to the new T_A which should be close to the value T_A that C has.
- second, see whether or not its time-stamp value T_C in the message is sufficiently fresh to be accepted.
- third, after successfully carrying out all these checks, it will accept T_A as the new time-stamp for A.

8.5.3 Time-stamp exchange: optimization of exchange and update

The problem with that protocol is that all the nodes should perform a handshake with all the other nodes, and in practice, if a node hears about a node of which it doesn't have a time-stamp, it should initiate an handshake.

This is not efficient, especially at network initialization. Hence the following optimization is proposed: 1) time-stamp handshakes are not performed by default, 2) the time-stamp update message from any other node is used to update all information about unknown time-stamps.

In more detail, this is how it works: remember that the time-stamp update message of A includes the time-stamp T_A of node A, (possibly) the time-stamp of the receiver node say, T_D and a list of time-stamps of other nodes, say T_E, T_F, etc .... Previously only T_A and T_D were used. This is because A might be compromised, and might have transmitted incorrect old values for T_E, T_F, etc ... Now, they are accepted with a "tentative" status, and retain that status, as long as no direct information is received (i.e. a time-stamp update message from originator E for setting the value of T_E, or a direct time-stamp handshake) - when proper information is received, the time-stamps are marked "authenticated".

An important point is that when no attack or network problems are present, then the time-stamp update messages received, both ways, will naturally perform the steps of the time-stamp handshake messages.

A mechanism is still needed to detect inconsistencies: a simple way is to use the normal time-stamp verification. When a message (for instance from E) is received by node A, the procedure is for A to check that the time-stamp of the message is consistent with the value that A has for the time-stamp of E. In our case, an inconsistency might be due to the fact that the known value of T_E, is a "tentative" value obtained from a compromised node. When there is such an inconsistency, and the time-stamp was "tentative", the solution is now to mark the time-stamp as "potentially attacked", to no longer use it, and to perform a direct time-stamp exchange handshake. The potentially attacked time-stamp is no longer used until the handshake is completed. If the handshake (with proper timeouts and retries) never succeeds, it might well be that it was a replay of a node that is no longer in the network, and the behavior is correct.

Note that, in any case, using this optimization, replays are accepted until the handshake is performed. It might well be acceptable, to accept temporary replays of OLSR protocol messages when there is a compromised node in the network. Note also, in any case, that such attacks are easily detected, after a time-stamp handshake, by all nodes which receive the incorrect time-stamp update messages ; and that the compromised node is easily identified: it is the originator of these messages.

This optimized algorithm results in a normal proactive exchange of time-stamps, complemented by a fall-back exchange which is reactive.

8.5.4 Time-stamp exchange: optimization for strictly increasing time-stamps

In the specific case where time-stamps are "strictly increasing", another optimization is possible. The greatest received (signed, as always) value of a time-stamp is always the "most correct".

Hence it is no longer necessary to send full "time-stamp update messages" and an alternative protocol is to have each node broadcast only its time-stamp $A \to all : \{T_A\}$ instead.

Because there are now no proofs of freshness in these messages, it is still necessary to perform a handshake once, to ensure that each node, from which shortened time-stamp update messages are received, is actually in the network. Otherwise it might be a replay.

The optimization of the previous section for time-stamp handshake is no longer correct, as some crucial information has been removed. One solution is, for instance, to first, when a time-stamp is tentative, to always update it to the greatest value ; second, to still send full "time-stamp update messages" at a slower pace. Other solutions include ideas like performing handshakes by default only with direct one-hop neighbors, to transfer the responsibility of checking that a node is actually in the network, to its one-hop neighbors.

8.6 Distribution of keys inside the distributed time-stamp protocol

It is interesting to use the distributed time-stamp protocol for another purpose: the distribution of keys.

Then the same protocol may be used, by sending keys along with time-stamps, enriching the previous messages with that information. As such key exchange is typically subject to man-in-the-middle attacks, two approaches could be used: either use an unoptimized node-to-node key exchange, or use a certification system.

In the first case, any algorithm from the literature for key exchange is used, resulting, in a network with N nodes, into NxN handshakes. Note that the advantage, is, of course, that symmetric keys can be used in this case. In the second case, we can assume that a certification authority, whose public key is known by all the nodes in the network, is signing all the public keys of the nodes of the network (along with the typical certificate validity duration). Then rather than just sending their public keys, the node will send their certificates instead - this allows other the nodes to verify that the node has been authorized in the network by the certification authority, and also prevents man-in-the-middle attacks.

9 Annex II: Centralized security architecture for OLSR with a PKI

This section describes a centralized security architecture, designed to reject all the intruders from the network. Based on the mechanisms described in section 4, we describe here how to implement a PKI (public Key Infrastructure) which will work hand in hand with OLSR and the basic security mechanisms defined in section 4.

9.1 Problem statement

The central hypothesis in this section is that there is a set of cryptographic capable nodes in the networks, and that none of these nodes are compromised. The goal of the architecture is to allow this set of cryptographic capable nodes to form a network amongst themselves and build routes to each other in a secure way.

The key issue is how a node will know and can be sure that another node will be a cryptographic capable node. To do that one must distribute keys which will allow a node (say, node A) to verify that another node (say, node B) is indeed a cryptographic capable node. It can be said that node B is a trusted node for node A.

In the next section, we present the basis of this security architecture. Following, we detail how these elements are linked and integrated with the OLSR protocol.

9.2 Fundamentals of the architecture

In this section, we describe the general design of the proposed security architecture, as well as detailing the individual components that make up the architecture.

9.2.1 Major design choices

We propose an architecture, using the following tools:

- *A*symmetric keys. In order to maintain the ability to admit or reject individual nodes at any time, individual asymmetric keys are used. This is detailed in section 9.2.3.
- *Centralized key distribution.* We assume a unique central node, a "signing authority", which knows and distributes the public keys of all the trusted nodes. It, alone, performs admittance control. While this assumption may seem contrary to the notion of "ad-hoc networks", we also observe that a number of applications seem to fall into the category of a centralized authority having the ultimate responsibility for admitting access to network resources: on a battlefield, for example, it is typically not desirable to have "just anybody" able to join the network, but devices which should be able to join are assigned centrally following the command structure in place.
- *Centralized time synchronization.* The nodes also maintain time synchronization by communicating with the central signing authority as in section 4.3

- *Proactive key and time distribution.* Following the philosophy of proactive construction of routes in OLSR, keys and time information are also distributed proactively. This information originates from the "signing authority".

In the following sections, we detail the components required to implement the design choices.

9.2.2 Subdivision into three classes of nodes

As a direct consequence of the centralized distribution of keys, not all nodes are aware of all the public keys of all the other nodes in the network. Thus, the architecture basically operates with three classes of nodes, as seen from an individual nodes point of view:

Untrusted Nodes
One node, A, considers another node, X, as an "untrusted node" if the public key of X is not known by node A or if the public key of node X is known, but not validated by a signing authority in the network, i.e. signature messages, received from an untrusted node, cannot be verified. Notice, that at network initialization, all nodes except the signing authority and the node itself will be "untrusted", from an individual nodes point of view

Trusted Nodes
One node, A, considers another node, X, as a "trusted node" if the public key of X is known by node A and the public key of node X is validated by a signing authority in the network, i.e. signature messages, received from trusted nodes, can be verified.

Signing Authority
The "signing authority" is a node, which has the special property that its public key is a priori known by all the nodes in the network. The signing authority has special responsibilities for the network, namely to distribute certificates and also to distribute base time one which each node synchronizes itself.

Our architecture is based on the assumption that trusted nodes entirely disregard traffic and information from untrusted nodes which aren't able to authenticate their control messages successfully. Then from the OLSR protocol's point of view, it is as if these nodes did not exist. As a result, no attack on the integrity of the information exchanges between the trusted nodes is possible, nor are attacks on the diffusion mechanism (MPR flooding) possible.

We will show later how in practice not *all* the traffic from untrusted nodes is ignored. In order to perform a successful network initialization, a limited set of information from untrusted nodes must be considered.

9.2.3 The Signing Authority: PKI

Our architecture is built upon a public key system, in which the signing authority diffuses the public keys for trusted nodes. The strength of public-key systems lies in the fact that public keys can be exposed to anyone - the difficult part being for a node to validate that a public-key, claiming to belong to a specific node, does in fact belong to that node. In our proposed architecture, this is ensured by the fact that the public key of the signing authority is known by all (trusted) nodes of the network. When issuing a message with public keys of the nodes in the network, the signing authority itself attaches its signature, i.e. the signing authority issues certificates for the trusted nodes.

In this context, the signing authority is performing the following tasks:

- it allows new nodes to register their public keys in a secure fashion (typically through manual authentication), whereby a new node becomes a "trusted node",
- it periodically distributes certificates, containing:
  - a list of public keys for all "trusted nodes",
  - a signature, verifying that the message, containing the public keys of the "trusted nodes" was, indeed, issued by the signing authority and was not modified while in transit.
- it periodically distributes the "base time", which is used by other nodes to synchronize themselves. It is signed with its private key.
- it answers the challenge-response protocol for "base time" acquisition. The answer is signed with its private key.

Each node wishing to participate in the network is required to register its public key with a signing authority. This is admittance. The signing authority periodically issues certificates. These certificates are broadcast to the entire network. Nodes, receiving the certificates, will store these for a specified amount of time, after which they expire. Hence, periodic refreshing of certificates is required.

9.2.4 Signing OLSR Messages and time synchronization

The mechanisms to sign OLSR messages and to synchronize network nodes described in 4.2,4.3,4.4 should be used with a system of public/private keys.

9.3 Preventing untrusted nodes in OLSR

Combined with the basic OLSR routing protocol, the components previously described can be applied to achieve the objective of admitting only trusted nodes into the ad-hoc network. We will describe the details of this in the following.

>From a network connectivity perspective, the primary task is to ensure that malicious link-state information (i.e. link-state information which, if introduced, would corrupt the image of the network topology) is not introduced into the network. In OLSR, this comes

down protecting the integrity of TC messages. TC messages are generated by nodes which are selected as MPRs, and are relayed by nodes which are selected as MPRs.

Thus, a node should ensure that it:

- selects only trusted nodes as MPRs,
- accepts to be selected as an MPR only by trusted nodes,
- forwards only broadcast/flooded messages from trusted neighbors,
- accepts only TC messages originating from trusted nodes.

If node A selects a node B as an MPR, node A effectively puts responsibility for advertising the link from node B to node A on node B. Only if node B is trusted can node A be certain that this responsibility is fulfilled. This corresponds to an instance of "incorrect traffic relaying" as described in section 3.

If node A is selected as an MPR for node B, node A assumes responsibility for advertising the link between node A and B. Only if node B is trusted can node A be sure of its identity and thus be sure that it is not introducing malicious link-state information into the network. This corresponds to an instance of "incorrect traffic generation" as described in section 3. In the same situation, node A accepts responsibility for forwarding broadcast/flooded traffic from node B and into the network. Unless node B is a trusted node, node A cannot be sure that node B will not generate and introduce excessive amounts of broadcast traffic which, when flooded to the network, may consume excessive resources and potentially prevent legitimate traffic from being transmitted.

If a node A accepts a TC message originating from a node B, node A assumes that the contents of that message are correct – and hence includes the information in its link-state database. Unless node B is trusted, node A cannot be sure that the information contained is correct. This, too, corresponds to an instance of "incorrect traffic generation" as described in section 3.

Thus, both the "Neighbor Sensing" and the general OLSR Message processing must be augmented to ensure network integrity. We will detail the required augmentations in the following.

9.3.1 Securing OLSR Message Processing

The simplest possible mechanism to keep untrusted nodes out of a network would be a simple rule stating: "a message, sent from an untrusted node, is silently discarded and neither processed nor forwarded". While simple, this condition is too restrictive, as will be illustrated below.

Let us imagine that all nodes required that all traffic they receive should be signed by a key they know (i.e. originates from a trusted node), with the proper time (i.e. the message is recent), before accepting them for processing and forwarding. This would lead to a signature deadlock problem on network initialization.

Upon network initialization, no nodes know any public keys, other than that of the signing authority (and, of course, their own). Thus, disregarding control traffic from the signing authority, all nodes will by default ignore control traffic from each other. Control traffic from the signing authority will be accepted by its neighbors since they know the public key of the signing authority in advance. The only node which may be selected as an MPR is, then, the signing authority itself. No other nodes will be selected as MPR, and no broadcast messages will be forwarded. In particular, all nodes 2 hops away from the signing authority will disregard control messages from nodes 1 hop away from the signing authority: the 2-hop nodes do not know the keys of the 1-hop nodes. The consequence is that links between 1-hop and 2-hop nodes will never be verified as "symmetric" and the signing authority will never select MPR's among its neighbors.

When the signing authority starts broadcasting certificates, these are received by the signing authority's neighbors – but are never forwarded since no MPR selection has taken place. Hence, the certificates never reach nodes beyond 1 hop from the signing authority, and no network formation takes place.

This is illustrated in figure 18 where node B rejects HELLOs from node A, thus stays asymmetrical for node A ; hence the signing authority has no MPR and its flooding stops to A. B never receives the keys of A (and other nodes).

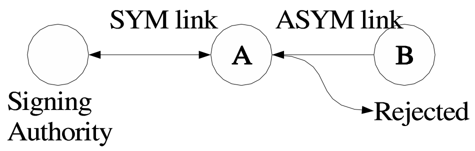

Figure 18: Signature deadlock example

Thus, to enable network initialization, the simple rule of "a message, sent from an untrusted node, is silently discarded and neither processed nor forwarded" is not applicable, and detailed considerations must be made.

9.3.2 Securing Neighbor Sensing

As we saw in the previous section, neighbor sensing must take into account the possibility of having a symmetric link with an untrusted neighbor in order to allow for certificate messages from the signing authority to propagate to all the nodes in the network. Returning to figure 18, we observe that the signing authority should select node A as an MPR in order for the certificates to reach node B. Considering the conditions listed at the beginning of this section, this is OK: node B is a trusted node by node A and vice versa. To allow for such an MPR selection, node A and node B must somehow establish that the link between them is symmetric, without node B selecting node A as an MPR etc ...

This is achieved by complying with the following considerations on HELLO message processing:

- A node must accept unsigned HELLOs from untrusted neighbors (i.e. neighbors, whose public key is not yet known to a node). Such HELLO messages are accepted under the restriction that:
  - asymmetric and symmetric links are considered as asymmetric and symmetric, respectively,
  - MPR links are considered as symmetric only (i.e. 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 neighborhood, perform MPR selection as specified.
- A node must, periodically, transmit HELLO messages, including the trusted neighbors (with status: asym, sym and MPR as appropriate) and untrusted neighbors (with status: asym, sym only)

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

With these rules, no signature deadlock is possible. Upon network initialization, the events are as follows: the signing authority will transmit its certificates and time, which is received by its 1-hop neighbors. Following a HELLO message exchange, the 1-hop neighbors will accept the untrusted 2-hop neighbors as symmetric (but not select MPR's 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 etc. The certificates will thus, upon network initialization, propagate from the signing authorities and towards the edges of the network. This is illustrated in figure 19

Notice that any information coming from "untrusted" nodes is only used to handle "untrusted" nodes: MPR selection etc is performed only among "trusted" nodes, as is MPR selector information only diffused about "trusted" nodes.

Also notice that no explicit mechanisms for revoking keys is presented. Since the diffused certificates have an associated expire time, they are naturally removed when they are not refreshed by the signing authority.

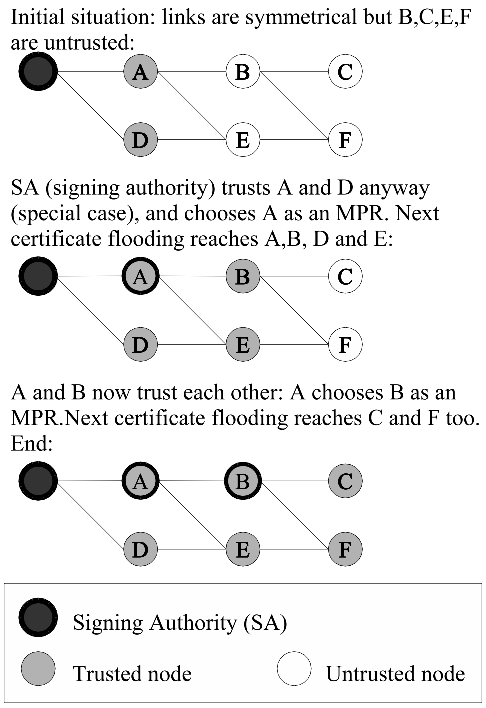

Figure 19: Adjusted processing example

10 Annex III: Signing algorithm

The aim of this annex is to specify the requirements that a signing algorithm must satisfy to be usable within the security architecture that we have designed to secure OLSR. We also review performances of well-known existing signing algorithms.

10.1 Requirements for a signing algorithm for OLSR control packets

We have mainly three requirements.

The first requirement is related to the computation requirement. OLSR is designed to work on network nodes with an average or even a small computation power. Thus the signing algorithm (signature and verification) must be able to be run with sufficient speed on such nodes. These nodes will probably have neither a lot of memory nor a dedicated processor or preprocessor.

The second requirement concerns the resistance of the signing algorithm. Of course if one can sign an arbitrary packets without knowing the signing key (equivalent to key retrieval), the signing algorithm is not efficient and can not be used. In such a case an intruder node will be able to forge any arbitrary control packets. If one can alter accordingly a message and its signature so that the altered message is still considered as a valid packet the signing algorithm can not be used. As a matter of fact an intruder will be able to forward altered control messages while the relayed control messages will still be considered as valid. The signing algorithm must resist such attacks.

The third requirement concerns the signature length. Of course a short signature is desirable, however a convenient tradeoff must be found between the signature length and the resistance of the signing algorithm.

10.2 Signature length of a few well-known signing algorithms

We consider the following signing algorithms:

- HMAC-MD 5,
- RSA,
- DSA,
- ECNR over GF(p),
- ECNR over GF(2^n).

We give in the table below the signature length of the previous algorithms.

Algorithm	**key: asym,sym**	**Signature length (bits)**
HMAC-MD 5	sym	128
RSA	asym	1024
DSA	asym	320
ECNR over GF(p)	asym	336
ECNR over GF(2^n)	asym	310

10.3 Performance of a few well-known signing algorithms

We report here on an excerpt from the benchmarks table of the Crypto++ 5.2.1 Library [11] a free C++ class library of cryptographic schemes maintained by Wei Dai. Benchmarks for the Pentium 4 2.1 Ghz processor were made on algorithms compiled with Microsoft Visual C++ .NET 2003 and run under Windows XP SP1. Benchmarks for the AMD Opteron 1.6 Ghz processor were made on algorithms compiled with GCC 3.2.2 and run under Linux 2.4.21. Times are in msec/operation, and signature length is in bits.

Operation	**P4 Time**	**AMD Time**	**Signature**
HMAC-MD5 operation over HELLO	0.27	0.38	128
HMAC-MD5 operation over TC	0.18	0.26	
RSA 1024 Signature	4.75	2.07	1024
RSA 1024 Verification	0.18	0.07	
DSA 1024 Signature	2.18	0.80	320
DSA 1024 Signature †	1.13	0.48	
DSA 1024 Verification	2.49	0.91	
DSA 1024 Verification †	1.79	0.78	
ECNR over GF(p) 168 Signature	3.34	1.49	336
ECNR over GF(p) 168 Signature †	1.90	1.04	
ECNR over GF(p) 168 Verification	6.31	4.03	
ECNR over GF(p) 168 Verification †	3.09	1.70	
ECNR over GF(2^n) 155 Signature	6.05	3.62	310
ECNR over GF(2^n) 155 Signature †	2.35	1.14	
ECNR over GF(2^n) 155 Verification	7.64	4.45	
ECNR over GF(2^n) 155 Verification †	4.06	2.02	

Schemes marked with the symbol † use precomputation; values are looked up in a table of 16 precomputed powers of each fixed base to speed up exponentiation. RSA uses 17 as the public exponent. Its implementation is based on the IEEE P1363 standard. DSA uses a 160-bit long value for q. Compared to the other algorithms in the list, the Crypto++ implementation of EC over GF(2^n) is less optimized. The implementation of ECNR follows the IEEE P1363 standard. HMAC-MD5 computations are made on an average HELLO and TC advertising 9 neighbors. HMAC-MD5 is significantly faster than signing algorithms using asymmetric keys.

11 Annex IV: Using IPSEC framework

In this research report the focus was on the issues of securing OLSR using the most direct methods. However, as IPSEC is an IETF standard for security, it is worth assessing the use of the IPSec framework within the security mechanisms we have designed to secure OLSR.

IPSEC and the related security working-group of IETF provide security by three complementary means:

- authentication (and optional encryption) of data packets.
- maintenance of "security associations", that is, of different keys of different connections.
- automated key exchange, and more generally PKI architecture and protocols.

The fundamental problems of IPSEC with respect to securing OLSR are the following:

- The automated key exchanges (which provide the automated time-stamp exchange, for replay protection), assume that the parties can reach each other. This is not the case in general with the OLSR protocol, because messages must first be authenticated, itbefore being accepted, and hence, a node which arrives in the network accepts no packets, and has no routes. Two remedies are possible, the first is to use pure flooding for the messages, the second is to change OLSR as we have shown in appendix 9.2.3.
- IPSEC protection, protects the packet itself, while the granularity of the protection that we proposed is the message. This has profound implications with respect to accountability if there is a compromised node. This is because many nodes repeat the messages that are diffused by MPR flooding (TC messages). 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, the only information deduced is that the compromised node is part of the (previously) trusted network. When the authentication is per message, the information deduced is that the origin of the faulty information is identified: it is the originator of the message.

  A remedy may be to forbid changing the packets: hence each message would go in a different packet ; this would have a certain cost on wireless networks, where MAC overhead per-packet is large in some cases – and as this might not be sufficient, other requirements, such as the use of tunnel mode, may be required, along with technical necessities such as use of the TTL field of the IP packets instead of the OLSR message.
- The current implementations support essentially symmetric keys. However this may change, as recent IETF drafts are available which propose asymmetric signatures, such as: *The Use of RSA Signatures within ESP and AH*, work in progress, Brian Weis, draft-ietf-msec-ipsec-signatures-02.txt.
- Managing a "group key" or a set of "group keys" in the context of a MANET, is a broader problem than just the issues studied at the IETF in the multicast or group

key management protocols (such as RFC 2093, *Group Key Management (GKMP) Specification*, or RFC 3830, *MIKEY: Multimedia Internet KEYing*, or Group management key exchanges), because in this case, all the nodes are senders and all the nodes are receivers (of the OLSR protocol messages), and also network splits or merges should be managed.

In general, a few IPSEC security schemes may be used, but using significant modifications (such as pure flooding for message transmission), at the cost of performance (for instance, if each node creates a session key to each other node, using pure flooding, the cost is N^3 where N is the number of nodes in the network), and security granularity. This would probably also result in using IPSEC protocols at the limits of their domain of applicability. And finally one should note that the complexity of IPSEC protocols is already greater than the complexity of the OLSR protocol itself, if the metric of "number of pages of the specifications" is used (for instance, IKE2 or GSAKMP have longer specifications).

Unité de recherche INRIA Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399