INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Detailed specifications of a security architecture for OLSR

Cédric Adjih, Paul Muhlethaler, Daniele Raffo

N° 5893

April 2006

Thème COM

Rapport de recherche

N 0249-6399 ISRN INRIA/RR--5893--FR+ENG

Detailed specifications of a security architecture for OLSR *

Cédric Adjih, Paul Muhlethaler, Daniele Raffo

Thème COM — Systèmes communicants
Projet HIPERCOM

Rapport de recherche n° 5893 — April 2006 — 30 pages

Abstract: In Mobile Ad Hoc Networks (MANETs), mobile nodes use wireless devices to create spontaneously a larger network, larger than the one hop radio range, in which communication with each other is made possible by the means of routing. The goal of this document is the study of security issue related to integrity of an ad hoc network. We only consider ad hoc networks using the OLSR [1] routing protocol. In a previous research report [2] we have carried out a theoretical analysis of this issue. In this document we aim at precising a detailed security architecture to protect the integrity of an ad hoc network using the OLSR routing protocol. We also validate this security architecture through simulations. This security architecture will be implemented in the demonstrator network that is deployed at the CELAR.

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)

Spécifications détaillées d'une architecture de sécurité pour OLSR

Résumé : Dans des réseaux mobiles ad hoc (MANETs), les noeuds mobiles utilisent le medium radio et des technonologies de routage pour créer des réseaux plus étendus que la couverture radio à un saut. Le but de ce rapport de recherche est d'étudier les problèmes de sécurité relatifs à l'intégrité d'un réseau ad hoc. Nous ne considérons que les réseaux utilisant le protocole OLSR [1]. Dans un précédent rapport de recherche [2], nous avons étudié ce problème d'un point de vue théorique. Dans ce rapport, nous précisons une architecture de sécurité détaillée utilisant le protocole de routage OLSR. Nous validons également cette architecture à l'aide de simulations. Cette architecture sera implémentée dans un démonstrateur réseau déployé au CELAR.

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

1 Introduction

Mobile Ad-hoc NETworks (MANETs) are infrastructure-free, highly dynamic wireless networks. Nodes 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 create connectivity within a MANET, a routing protocol must be used.

Currently, two complimentary classes of routing protocols exist in the MANET world. Reactive protocols (e.g. AODV [3] and DSR [4]) acquire routes on demand through flooding a "route request" and receiving a "route reply". 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 [1] and TBRPF [5].

The goal of this document is the study of security issue related to the integrity of an ad hoc network. We only consider the use of the OLSR [1] routing protocol. In the previous deliverable [2], we have carried out a theoretical analysis of this issue. In this document, we aim to precise a detailed security architecture for the demonstrator network and to quantify the behaviour of this security architecture.

1.1 Summary of the results of the theoretical study

In the theoretical [2] study we have identified for OLSR five attacks towards the network 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 the network connectivity.

In this theoretical study we have introduced the concept of *a cryptographic capable node*. Such a node has received valid keys and which can sign and authenticate the control messages.

In an ad hoc network where the nodes meet are cryptographic capable, a node is said to be compromised if it does not process and emit control traffic in 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.

Under the assumption that there is no compromised node in the network, cryptographic capable nodes can counter all the identified attacks except the relay attack. They have to use signature and time-stamps mechanisms. Doing so, will prevent intruder nodes to be part of the network. The relay attack can be countered in most of the situations when the network nodes know their own location.

If we assume that there are compromised nodes in the network, securing OLSR is much more complex. The previous techniques that prevent an intruder nodes to be part of the network are not operating. The general idea is, in such a case, to detect compromised nodes or compromised links and to remove such nodes or links. A perfect securisation under such an assumption seems out of reach.

1.2 Document outline

The remainder of this paper is thus organized as follows: section 2 presents the basic security mechanisms which will be integrated in the demonstrator. Section 3 presents the detailed specifications of the security architecture that is implemented in the demonstrator. In this section, we discuss the reasons of the modifications we made to security architecture presented in the theoretical study [2], the format of the security information required for the security architecture and the implementations issues. We also precise what are the required modifications to the specifications of OLSR to implement the proposed security architecture. Section 4 presents the scenarios where intruder nodes try to launch attacks against the network integrity. We study in this section the network behaviour when the network is attacked. We also quantify the behaviour of the network when the network uses the proposed security information. In the annex section 6, we detail the modifications to the OLSR RFC 3626 required to implement the security architecture. Since the security architecture proposed in this document uses the nodes' clock to generate time-stamp it requires high quality clocks or a synchronization algorithm. We propose in the annex, a protocol OSTP (OLSR Secure Time Protocol) to generate timestamps with non synchronized clocks.

2 Overview of the security architecture

In the theoretical study, we have studied how to counter attacks when there are compromised nodes in the network.

We have shown that various techniques could be used to counter attacks in this situation. To counter incorrect control message generation we have shown that link signatures where both end point nodes sign the message can be used. To counter bad message relaying attacks, techniques based on flow conservation can be used. We have shown that these techniques allow one to counter several configurations of attack but fail to counter all the possible attacks. Countering attacks with compromised nodes implies to use complex techniques, additionally the each attack has a different parade. Moreover not all the attacks can be handled.

The difficulty to handle situations with compromised nodes has lead us to favor the assumption of an ad hoc network without compromised node. A possible solution to achieve this assumption could be to protect network nodes with a hardware protection. For instance, we could assume that if a node falls under the control of a foe, this latter will be neither able to change its behaviour nor to know the node cryptographic key.

In the following we will thus only consider the assumption where there is **no compromised node** in the network.

If one decides to use asymmetric keys, the public keys must be distributed in the ad hoc network. In the annex of the theoretical study, we have studied how this distribution can be handled by a centralized entity. We have shown that this distribution leads to modification of OLSR. We consider that the implementation of these changes although appearing as small changes in the OLSR specifications is rather difficult.

The solution for the security architecture of demonstrator will use a shared key with symmetric cryptography.

As in the theoretical study, our design security architecture will be based on the two main mechanisms:

- **S**ignature mechanism
- **T**ime-stamp mechanism

2.1 Signature mechanism

In the designed security architecture for the demonstrator, we use the signing mechanism that has been introduced in the theoretical study. However the signature format was changed and additional protection was introduced.

2.2 Time-stamp mechanism

In the designed security architecture, we keep the time-stamping mechanism. However for the designed security architecture we will use a simpler time-stamp mechanism than all the ones described in the theoretical study.

2.3 Signing algorithm and key distribution

For the demonstrator, three possibilities were considered: authentication with symmetric cryptography, traditional asymmetric cryptography or identity-based (pairing-based) cryptography.

As shown in the theoretical study, we have seen that using asymmetric keys (with traditional cryptography) requires the distribution of these keys: this leads to overhead and additional attacks.

Identity-based cryptography (based on pairing) could be an interesting solution, however the signature and verification times are beyond the computational power of the routers (see [6]).

In the demonstrator, we will therefore implement the HMAC authentication algorithm (which MD5 hashing function) for signatures. This technique uses a symmetric shared key.

3 Detailed specifications of the security architecture

We are now giving the detailed specifications of the security architecture. We first discuss what are the packets that must be signed. Then, we study how these packets must be signed. Lastly, we study how time-stamps are generated and optional features of the security architecture.

e are detailing the time-stamps mechanism.

3.1 Security of HNA and MID messages

In the theoretical study [2], we have proposed to authenticate HELLO, TC control messages by using signatures.

However, there exist other messages which carry routing information for standard OLSR: the MID and HNA messages.

Signing HNA messages is mandatory to protect the network against dissemination of wrong accessibility information. Similarly, when the network includes nodes with multiple interfaces, MID messages are used to get the correct routes to all the network interfaces but also to correctly compute the MPR set. Thus signing MID is necessary to prevent attacks which can be very damageable.

In the demonstrator we will thus sign Hello, TC, MID and HNA messages.

3.2 Format of the signed messages

In the theoretical study [2], for each control message (HELLO, TC) generated, a corresponding SIGNATURE message is generated. The control message and its signature can be sent in different packets if needed. SIGNATURE messages are used by a receiving node to authenticate the corresponding OLSR control message.

3.2.1 Format of signature messages in [2]

The SIGNATURE message is shown in Figure 1. The message carried a `MSN Referrer` field in order to identify an one-to-one correspondence between a control message and its SIGNATURE message.

The `Sign. Method` field specifies which method, among a predefined set, is being used to generate the signature. This may include information about keys, cryptographic functions, and time-stamp methods. The `MSN Referrer` field of the SIGNATURE message contains the value of the `Message Sequence Number` of the control message to which this signature is associated, see Figure 2. The correspondence achieved by the Message Sequence Number is unique only if possible wraparound of the 16-bit field is disregarded; however this is not a problem, since a node uses further signature verification to check the correspondence between the control message and the signature message.

The approach implemented in the theoretical study makes unnecessary to send the SIGNATURE message and its associated control message in the same packet, as the messages could be reordered and re-associated later. Also another advantage of this design is the signature message is introduced as an orthogonal feature, and the compatibility with standard OLSR nodes (without security features implemented) is kept.

However, another approach was favored, mainly because of **implementation facility** and **robustness**. Precisely, when the signature is sent in a separate control message, every node needs to store the received messages (control or signature messages, whichever arrives first) in a buffer. This requires more system resources and is more prone to failure and Denial-of-Service attacks.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 5893 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Sign. Method  |    Reserved   |         MSN Referrer          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            Time-stamp                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                            Signature                          :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 1: SIGNATURE message format as proposed in the theoretical study.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 5893 0 1

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Sign Msg Type |     Vtime     |         Message Size          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Originator Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time To Live |   Hop Count   |    Message Sequence Number    |\
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|     Message Content Size      |    Reserved   | Message Type  | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | Link
|                                                               | | between
:                            MESSAGE                            : | the
|                                                               | | message
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | and its
                                                                  | signature
                                                                  | message
                                                                  |
                                                                  |
                                                                  |
                                                                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
| Sign. Method  |    Reserved   |         MSN Referrer          |/
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Time-stamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
:                           Signature                           :
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 2: The controle message and its signature message.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Sign Msg Type |     Vtime     |         Message Size          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      Originator Address                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Time To Live |   Hop Count   |    Message Sequence Number    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+\
|     Message Content Size      |    Reserved   | Message Type  | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | Control
|                                                               | | Message
:                            MESSAGE                            : |
|                                                               | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+/
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+\
|   Security Information Size   |    Reserved   | Flags | Meth. | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                     Optional  Time-stamp                       | | Security
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ | Infor.
|               Optional Source Interface Address               | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |
|                                                               | |
:                           Signature                           : |
|                                                               | |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+/
```

Figure 3: Signed message format.

3.2.2 Signing scheme adopted in the demonstrator

In the detailed specifications of the security architecture, the format of the signatures is modified: the control message header, control message content and the signature are sent together inside a same OLSR message.

The current (and implemented) version of the signed OLSR message is based on the following format represented on figure 3. Mainly we have an usual OLSR control message followed by Security Information fields.

The `Signed Message Type` field is set to the corresponding value of the signed control message:

- a HELLO_MESSAGE becomes a SIGNED_HELLO_MESSAGE
- a TC_MESSAGE → SIGNED_TC_MESSAGE
- a MID_MESSAGE → SIGNED_MID_MESSAGE

- a HNA_MESSAGE → SIGNED_HNA_MESSAGE

All the other fields are set to the value of the control message fields which is being generated (Vtime, TTL, hop count, MSSN, ..) as in the OLSR specifications (RFC 3626).

The format introduces a "Security Information" part, which includes:

- Security Information Size (2 bytes) which gives the length (in bytes) of the Security Information
- Flags (4 bits): flags which indicates, whether or not the optional time-stamp field, and the optional "source interface address" is present.
- Method (4 bits): the signature method used.
- Time-stamp (4 bytes, optional):
- Source Interface Address (4 bytes, optional):
- The signature field.

The `Time-stamp` field contains the time-stamp itself, measured in seconds. The time-stamp is 32-bit long and represents the standard Unix time, which is encoded in a 32-bit signed integer data type. The Unix time measures the time elapsed in seconds since 00:00:00 UTC on January 1, 1970.

The `Source Interface Address` field contains the Source Interface Address of the control message. This optional field is only useful for Hello packet. Section 3.4 explains the use of the optional features.

The `Signature` field contains the signature, computed on the sequence of bytes made from the whole message, except (naturally) the Signature field (see figure 3) and with the `Time To Live` and `Hop Count` fields set to 0. These fields are reset in the signature computation because they are modified while the message is in transit, and subsequently the signature of the message would be invalidated.

The difficulty posed by handling control messages and their appended security information is solved as follows. The control message may be fragmented if necessary, so that control message and its related security information is smaller or equal than the MTU of the network. If the control message is fragmented, an independent SIGNATURE message must be computed and assigned to each fragment. Fragmentation may also be used for messages that are awaiting in the relaying queue, in order to insert these messages in the packet ready to be sent. Although optimizations could be possible, we always consider the network MTU (i.e. the minimum bound of the MTUs of all the link pairs in the network) to fragment control message if needed.

3.3 Time-stamps

The time-stamps are simply the times given by **'nodes' internal clock**. A strict synchronization of nodes' clocks is not necessary since the time-stamp is used to complete the already existing protection offered by the Message Sequence Number and the duplicate set. As a matter of fact, messages that are already in the duplicate set are silently dropped.

The criterion to verify whether a time-stamp is stale is $|$ `Time-stamp` $-t_0| \leq \Delta t$, where t_0 is the current time at the receiving node and Δt is the accepted value for synchronization discrepancy, also called below time-stamp tolerance. Since the holding time in the duplicate set is by default 30 seconds, we can use $\Delta t = 15$ seconds as a default value. Replay attacks of messages less than 15 seconds old will be avoided by the duplicate check of OLSR. This check is actually in the processing condition of a message.

If the 'nodes' clocks are of very poor quality, it is still possible to use them to generate time-stamps. In the annex , section 6.1, an "OLSR Secure Time Protocol" (OSTP) is presented. It allows nodes to run with non-synchronized clocks while the timestamps are still using the nodes' clocks.

3.4 Optional features

We have found an additionnal relay attacks than the one proposed in the [2] that is the reason why we have introduced optional features in the signature information field.

The first attack[1] consists in changing the value of the `Time To Live` and `Hop Count`. As previously said, the `Time To Live` and `Hop Count` fields in the message headers cannot be included in the signature computation, since these fields change at each hop of the message and this would interfere with the correct verification of the message by the receiving node. However, this leaves the door open to an attack where an adversary relays tampered messages whose TTL has been set to 0 or 1 or, more generally, to a lower value than the original.

The second attack consists in sending an Hello message sent on a given interface and to replay it on another interface. This is possible since in an Hello control message, the interface address is not in the control message but must be found in the IP header.

The first weakness can be patched by ignoring the `Time To Live` field, and referring to the `Time-stamp` field (which is protected by the signature) to limit the forwarding radius of the message.

The second weakness can be avoided by using the `Address` field. This field will hold the source interface address of an hello message. Since this field is included when the signature is formed, it is impossible, for an attacker, to change the source interface address of an hello message without invalidating the signature of this hello message.

3.5 Modification to the standard OLSR protocol

Implementing the proposed security architecture can be done re-using most of the OLSR specifications. This requires to modify a few parts of its basic functioning. We are presenting them below.

[1] Actually this first attack was already mentionned in [2]

3.5.1 Sending a signed control message

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

1. the node generates the control message;

2. the node creates a new header with the content of the message.

3. the node creates a security information field:

   - the node retrieves the current time, and writes it in the `Time-stamp` field (optional);
   - the node retrieves the source interface address, and writes it in the `Source Interface Address` field (optional);
   - the node computes the signature, and writes it in the `Signature` field;

4. the node puts the control message followed by the security information fields in the packet, in this exact order.

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

The signature is computed on the sequence of bytes made from the whole message, except (naturally) the Signature field (see figure 3) and with the `Time To Live` and `Hop Count` fields set to 0. These fields are reset in the signature computation because they are modified while the message is in transit, and subsequently the signature of the message would be invalidated

3.5.2 Receiving and checking a signed control message

The Duplicate Set of the standard OLSR is used as it, but the processing is modified in order to add security checks

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

1. the node checks the processing condition of the p.16 of the RFC 3626.

2. the node checks the validity of the security information of the control message: the time-stamp (if required by the type of control message), the interface source address (if required by the type of control message), and signature.

3. if one of the previous verifications fails, the processing of the control message will stop (and the message is ignored): no duplicate tuple is created for that message (or modified is one existed beforehand).

Message	Message Type Value	Time-stamp?	Interface Address?
`SIGNED_HELLO_MESSAGE`	0xcc	mandatory	mandatory
`SIGNED_TC_MESSAGE`	0xcd	mandatory	no
`SIGNED_MID_MESSAGE`	0xce	mandatory	no
`SIGNED_HNA_MESSAGE`	0xcf	mandatory	no

Table 2: Proposed values for signed messages

Flagtype	Flag Value	Applicability
`FLAG_DEFAULT`	0	Hello,TC,MID,HNA
`FLAG_WITH_TIMESTAMP`	1	Hello,TC,MID,HNA
`FLAG_WITH_ADDRESS`	2	Hello

Table 3: Proposed values for flags

4. if the previous checks were satisfied, the control message is accepted and processed following the rules of the standard OLSR. If not, the control message (including the security information) is dropped.

The signature is verified according to subsection 3.5.1.

As an optimization, the verification of the security information may delayed until the node is certain to perform some steps in the processing of the mesage (such as message content processing, forwarding, or update of duplicate table).

3.5.3 Proposed Values for Constants

Four message types are introduced for security. The specification of the value to be set in the "Message Type" field and whether or not some optional fields are demanded, is given on table 2.

Three flag types are introduced. The specification of the value to be set in the "Flags" field and applicability of these flags is given on table 3.

The HMAC authentication algorithm is referenced by

- `AUTH_METHOD_HMAC` = 2

The time-stamp tolerance used in the simulations presented in section 4:

- `TIMESTAMP_TOLERANCE` = 15 seconds

4 Validation of the security architecture

4.1 Presentation of the scenarios used for the validation

As already stated, we are not not handling attacks when the network comprises compromised nodes. We will use four scenarios to validate the proposed security architecture. There is no attack in the first scenario.

The three other presented scenarios correspond mainly to attacks related in the theoretical study as incorrect control message generation attacks. We plan to qualify the performances of the network under these attacks on the following points :

- correct operation, in other words, is the attack successful or not?
- what is the cost for the network to use the security architecture that we have designed.

4.2 Scenario 0

There is no intruder in this scenario. The aim of this scenario is just to validate that a network using the defined security architecture is working properly.

4.3 Scenario 1

In this scenario we just suppose that the intruder nodes do not sign their control packets. The goal of this scenario is to verify that in such a case, intruder nodes do not take part to the network. Thus the attack is unsuccessful. Since intruder nodes are not admitted in the network, they can not either launch successful bad data traffic relaying or bad control traffic relaying attacks.

4.4 Scenario 2

In this scenario we just suppose that intruder nodes do not sign their control packets properly; they use a wrong key. The goal of this scenario is to verify that in a such case, the intruder nodes do not take part to the network.

4.5 Scenario 3

In this scenario, intruder nodes launch replay attacks. To simplify the construction of this scenario, we just suppose that the replay attacks are equivalent to intruder nodes with a de-synchronized clock.

4.6 Qualification of the performances of the network

4.6.1 Overview of the methodology

In order to check the proper functionning of the specification and the implementation, several validation tests were performed using the version of OLSR with security.

All the validation tests presented here were performed with simulations. In fact, OOLSR has the ability to perform simulations with several nodes, using a simplified model of transmission (no collision, infinite bandwidth), designed to validate the functioning at the protocol level.

In addition, because the security architecture proposed in this document does not modify the parts of the implementation related to operating system interaction (setting routes, sending and receiving messages, ...), these tests would actually validate most of the changes for security.

The simulation scenarios are using nodes spread inside a flat square area, with dimension 1x1 (km).

4.6.2 Validation with scenario 0

In this section, a given scenario is used to verify the proper functionning of the secured version of OLSR: the same scenario is run first without enabling security (hence the nodes exchange standard OLSR messages) and second, by enabling security (hence the nodes exchange signed OLSR messages).

In the four scenarios, 100 OLSR nodes are randomly placed on the area, the radio range is set to 0.15 (km), and the simulation is run for 300 seconds, without any motion.

Some results of the first and second scenarios are displayed on figures 4 and 5: the 100 nodes are the black circles. One of the nodes is selected (the node in red), and all the routes from this node to all the other nodes are displayed. More precisely, all the links used in those routes are displayed as blue links.

As we can see, a route exists from the red node to all the others, either when the simulation is performed with security, or without security. Note that although the routes are slightly different in a few cases because of divergent random number generation in the simulation, they are alternate shortest path routes in both cases.

Looking in the log files of the simulation, it was verified that signed messages were actually sent, by verifying information such as:

```
    1.338537989 [security] n0 M[n0:1] added-auth-info full-message=(cc 86 00 68 00 00 00 00 00 00 00 01 00 40 01 01 00 00 05 03 01 00 00 38 16 00 00 00
41 00 00 00 07 00 00 00 4c 00 00 00 49 00 00 00 0f 00 00 00 13 00 00 00 35 00 00 00 04 00 0 0 00 3c 00 00 00 19 00 00 00 30 00 00 00 4b 00 00 00 00 1c
02 32 00 00 00 01 00 00 00 00) auth-info=(e0 a1 29 88 76 ce 25 96 7b 12 85 38 36 f2 a5 65)
```

which indicates that the authentication information (HMAC-MD5) field was added to a generated message in signed format (here the first byte of the message is `0xcc`, hence it is a SIGNED_HELLO).

Overall, this validates several points of the implementation including the following features:

- Proper formatting of OLSR messages with signature.

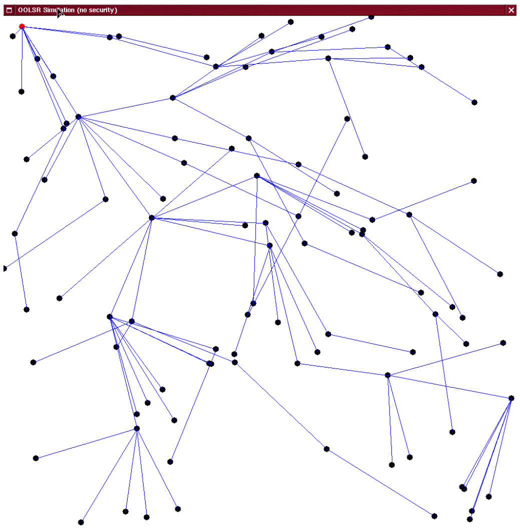

Figure 4: Simulation without secure messages

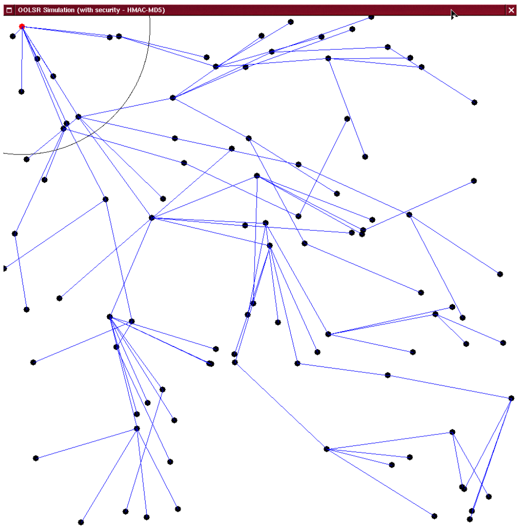

Figure 5: Simulation with secure messages

- Proper parsing of OLSR messages with signature.
- Proper signature of OLSR messages (verified in the log files).
- Proper processing of signed OLSR messages.
- Proper forwarding of signed OLSR message (TC messages).

4.6.3 Evaluation of the overhead

The overhead was evaluated with previous scenario, comparing some results obtained, with security on one hand, and without security on the other hand.

In addition, on the same topologies as previously, another set of simulations was run: again, with and with security, but with a radio range set to 0.5 (km); their results are displayed on figures figures 6 and 7.

The results are summarized on the following table.

Since use of security information to messages adds 20 bytes (TC/HNA/MID messages) or 24 bytes (HELLO messages), in IPv4 and for HMAC-MD5, we see additionally that most of the packets are including only one message in the table in the case of lower density with range 0.15 (km).

Exploiting the information on the increase of the average packet size, we can easily compute the overhead introduced by the security architecture. If we measure this overhead in term of number of bytes sent the security architecture adds about 57 percent of control

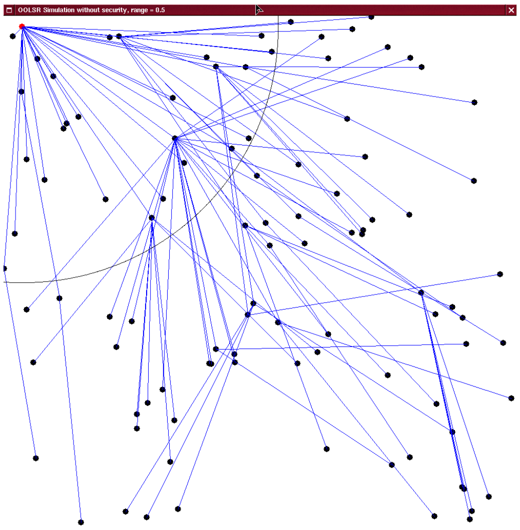

Figure 6: Simulation without secure messages, range = 0.5

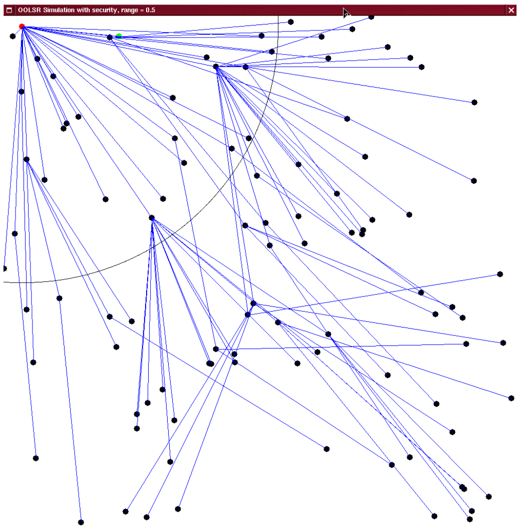

Figure 7: Simulation with secure messages, range = 0.5

Simulation Parameter	Standard OLSR	Secure OLSR
Number of routes	9900	9900
Nb of packets sent	81105	81017
Nb of bytes sent	4663756	7306380
Avg. packet size	57.5 bytes	90.2 bytes
Addition Overhead	reference (=0 %)	56.8 %

Table 4: Global results in simulation with and without security, range = 0.15

Simulation Parameter	Standard OLSR	Secure OLSR
Number of routes	9900	9900
Nb of packets sent	22404	22377
Nb of bytes sent	3704100	4394980
Avg. packet size	165.3 bytes	196.4 bytes
Addition Overhead	reference (=0 %)	18.8 %

Table 5: Global results in simulation with and without security, range = 0.5

overhead (for IPv4) in the case of low density, and 19 percent overhead in the case of higher density. Hence, the additional overhead is reasonnable. It should be noted that when control traffic becomes important because density increases, the message sizes increase, and the relative overhead due to security is lowered, as illustrated by the simulations with range = 0.5 (km).

4.6.4 Validation with scenario 1

In this section, the scenario 1 is simulated. 100 nodes are spread on a square, as before in section 4.6.2. The difference here, is that one part of the nodes are running the secure version of OLSR (hence exchange signed OLSR messages), while the other part is using standard OLSR (without signed messages).

Precisely, all the nodes in the left part of the area (which have x coordinate $<$ than 0.5) are using secure OLSR, while the right part is using standard OLSR.

Some results of the simulation are displayed on figures 8 and 9: the routes to all nodes as seen by one node of each part, are displayed. On the graph, we can confirm that indeed,

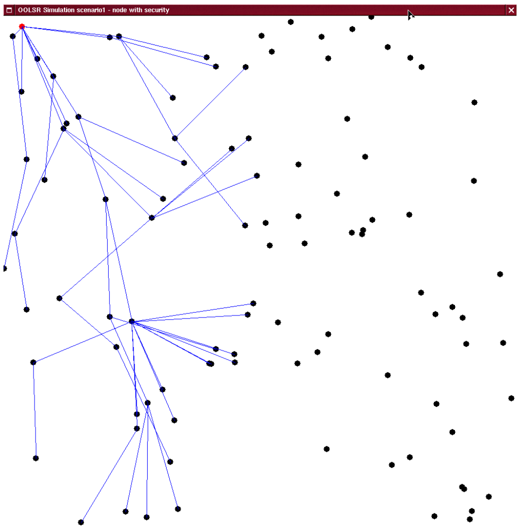

Figure 8: Routes of a node with signed messages

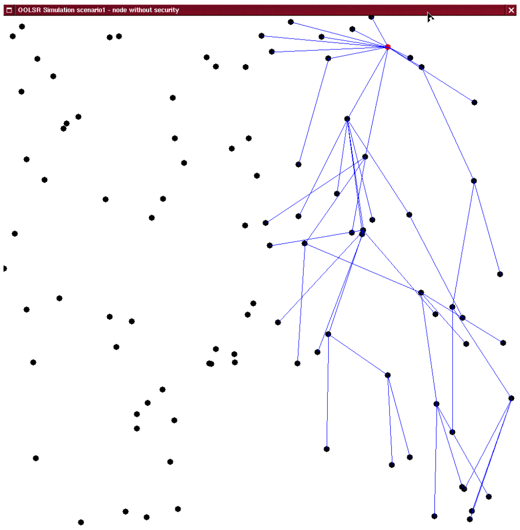

Figure 9: Routes of a node with standard messages

no node of the right part (without proper authentication), was allowed to participate in the network of the nodes with security.

It was further manually verified that the routes of every node of one part, never include or reach any node of the other part.

The results indicate the proper behavior of the implementation.

4.6.5 Validation with scenario 2

In this section, the scenario 2 is simulated. 100 nodes are spread on a square, as before in section 4.6.2. The difference here, is that one part of the nodes are using one key, while the other part is using another key for authentication (signing and verifying messages).

Precisely, all the nodes in the top part of the area (which have y coordinate < than 0.5) are using the symmetric key "sym-key1", while the bottom part is using the symmetric key "sym-key2".

Some results of the simulation are displayed on figures 10 and 11: the routes to all nodes as seen by one node of each part, are displayed. On the graph, we can confirm that indeed,

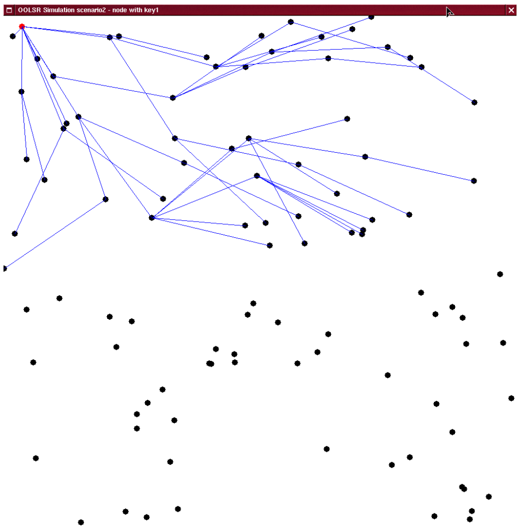

Figure 10: Routes of a node with first key

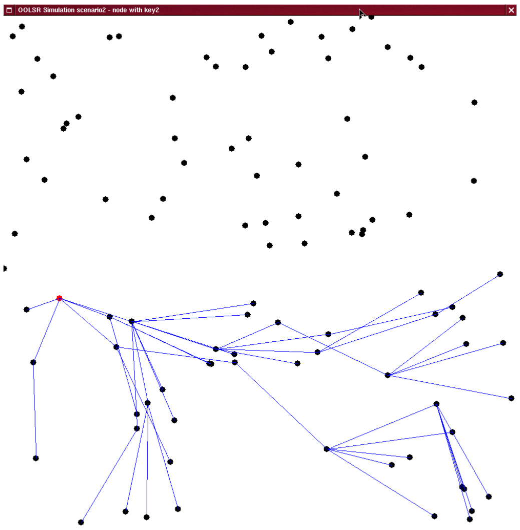

Figure 11: Routes of a node with second key

no node of the bottom part (without proper authentication), was allowed to participate in the network of the nodes of the top part, and conversely.

It was further manually verified that the routes of every node of one part, never include or reach any node of the other part.

The results indicate the proper behavior of the implementation.

4.6.6 Validation with scenario 3

In this section, the scenario 3 is simulated. 100 nodes are spread on a square. The difference here, is that one part of the nodes are running a clock which identical to the clock of the simulator, while the other part have a clock which is off by 100 seconds.

Precisely, all the nodes in the top right part (which have y coordinate $<$ than x) have same time as the simulator, while the bottom left part have a time which is off by 100 seconds.

Some results of the simulation are displayed on figures 12 and 13: the routes to all nodes as seen by one node of each part, are displayed. On the graph, we can confirm that indeed,

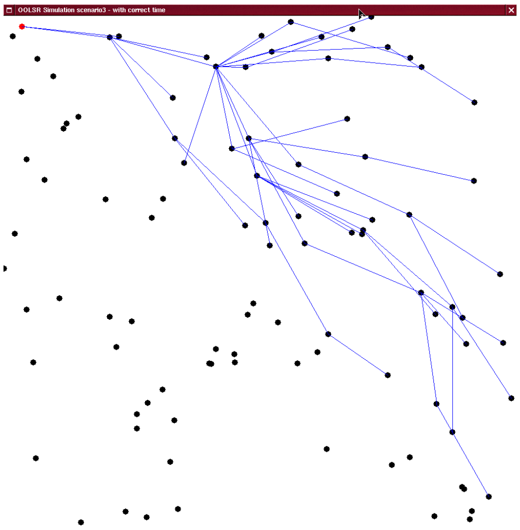

Figure 12: Routes of a node with clock offset=0

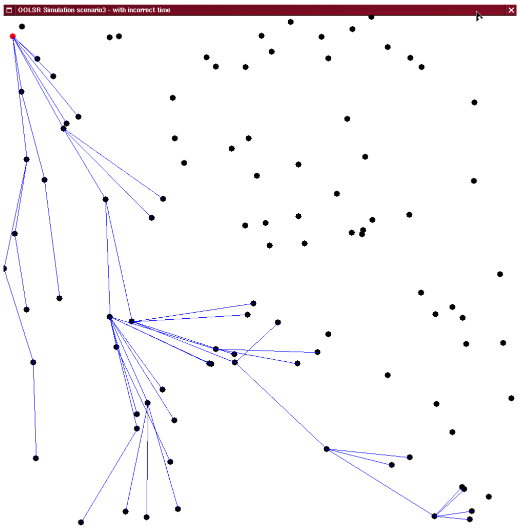

Figure 13: Routes of a node with clock offset=100

no node of the bottom left part (without proper authentication), was allowed to participate in the network of the nodes of the top right part, and conversely.

It was further manually verified that the routes of every node of one part, never include or reach any node of the other part.

Now, an additional experiment is to run the same scenario, while disabling the verification of the timestamp in the implementation: the messages are still signed and still include timestamps, but the implementation no longer checks that the timestamps are correct. Some results of the simulation under those conditions are displayed on figures 14 and 15: the routes to all nodes as seen by one node of each part, are displayed. We see that in the current case, (since the nodes have the same keys), indeed, all nodes can reach all other nodes (hence, note that in these conditions, the nodes are vulnerable to replay attacks).

The results indicate the proper behavior of the implementation.

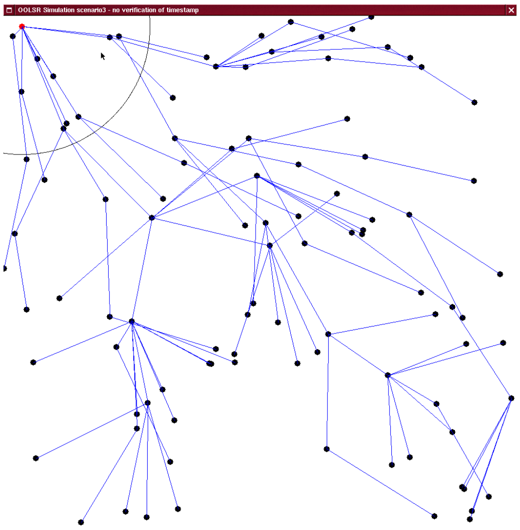

Figure 14: Routes of a node with clock offset=0 and no timestamp check

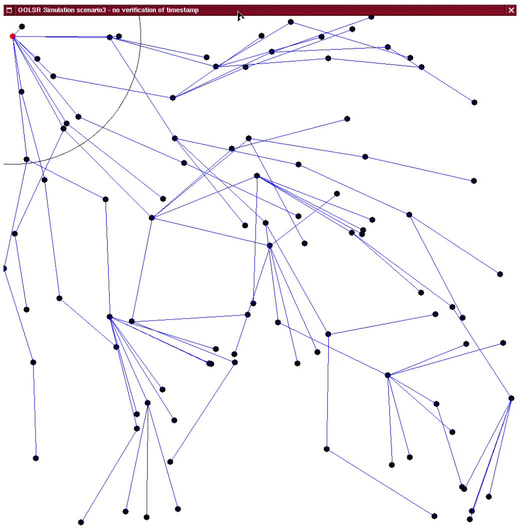

Figure 15: Routes of a node with clock offset=100 and no timestamp check

5 Conclusion

In this report we use the assumption of network without compromised nodes. As a matter of fact, although there are hints in the theoretical study [2] to counter possible attacks with compromised nodes in the network, a perfect level of security seems out of reach.

Key distribution seems also a quite difficult task in ad hoc networks as identity based signatures (based on pairing algorithms) requires too much resources to be implemented in OLSR nodes with the present computation power of embedded systems. For this reason, we use a symmetric shared key to sign OLSR messages and the widespread HMAC autentication algorithm.

We give the detailed specification of a security architecture for OLSR. We explain the reasons for the small changes that we did from the basic security architecture proposed in [2]. For **implementation facility** and **robustness** reasons, we use a signature scheme where the signature information is put just after the control message. The control message and the related signature information forms a new "kind" of control message.

We validate the implementation of the security architecture on test scenarios. We show, by simulation, that the implementation operate approprietly and that the security architecture can successfully counter attacks. The overhead generated by the security architecture is also evaluated.

6 Annex

6.1 Auxiliary functioning: OLSR Secure Time Protocol

In the document, it was assumed that the different nodes participating in the MANET are using their clock to timestamp the different messages. This is possible when nodes possess high quality clocks, or are synchronized regularly. Then the difference between nodes' clocks remains limited. This is usually the case.

If this assumption can not be met, and for the completeness of this document, an "OLSR Secure Time Protocol" (OSTP) is presented in this section. It allows nodes to run with non-synchronized clocks while the timestamps are still using the nodes' clocks.

6.1.1 Overview

OLSR Secure Time Protocol can be seen as a derivative of the "distributed time-stamp protocol" described in the theoretical study, and of the one, more complex, in [7]. However, there is an assumption used in this document: the clocks are used for timestamping, and all of them are non-decreasing (an assumption compatible with the fact they are not synchronized). In this context, it is no longer necessary to use an authentication handshake, such as the corrected Needham-Shroeder protocol, which included a small "reactive" component. As a result, the "OLSR Secure Time Protocol" can be now a transposition of the mechanisms of the NTP protocol [8] (and RFC 958 [9]).

Let's recall the basic idea behind the NTP protocol: two nodes A and B that are attempting synchronization, exchange time messages with:

- The current time T_3 (time of generation of the message)
- The time at which the previous message from the other machine was received T_2
- The time of generation indicated inside that previous message, T_1

When a machine receives such message, it records the current time T_4. Then assuming that the two machines have a constant time difference ΔT, the following basic inequality is verified:

$$T_1 - T_2 \leq \Delta T \leq T_4 - T_3 \tag{6.1}$$

This inequality gives one interval for ΔT. The range of the interval correspond to one round-trip time (hence infinite transmission speed would result in interval a range of 0, i.e. an unique precise value). Because such messages are periodically transmitted, the NTP protocol accumulates them and perform some data-filtering. After large scale experiments on ARPANET and NSFNET [10] with hundred of thousands of nodes, the NTP authors have chosen to use data-filtering using a *minimum filter*: from all the recent intervals, the one with the smallest range is used. Further processing is done, as NTP performs the additional task of synchronizing clocks of the hosts.

On the security of such messages, if we assume authentication, the issue is replay attacks: as noted in NTP, T_1 provides a proof of freshness for the receiver, and T_3 can be also used to eliminate replays.

In OSTP, the same ideas are used, with some changes:

- Each node generates one STP message intended for all nodes of the network (instead of one NTP message for every other node of the network), in the spirit of our previous proactive protocol in [7], including all the time information for all the nodes.
- In the same spirit, the clocks are not synchronized, but the difference with the clock of other nodes in maintained. It is used when checking time-stamps.
- $T_1 - T_2$ is not computed on the receiver side: instead the sender transmits its estimate of the upper bound (his $T'_4 - T'_3$), which gives a lower bound for the receiver side.
- STP messages are authenticated, as the STP messages have the option: however they are authenticated using the OLSR "Signed message format" described in this document.
- Some interesting security features of NTP are used, sometimes slightly altered.

6.1.2 STP message format

The STP messages have the format of standard OLSR messages and must be using the signed message format. The content of the message has the format represented on figure 16. The times are coded on 64 bits, in a similar way with earlier NTP versions, with the integer part coded as a signed integer, and a fractional part coded on 32 bits.

The "Time Difference" field is actually optional and its format is slightly different: its fractional part is coded on 31 bits, and the last bit merely indicate if the value of this field is present or not (i.e. should be taken into account or not).

6.1.3 Time Protocol Information Set

Each node implementing OSTP maintains a "Time Protocol Information Set", a set of "Time Protocol Tuples" which include each the following information:

- `T_peer_address`: the address of another node
- `T_peer_last_time`: records the "Generation Message Time" in the last message received from that node.
- `T_offset_lower_bound_list`: a list of couples `(time, expiration time)` which is the list of the ΔT described previously, computed on the last received messages from that node.
- `T_has_offset_upper_bound`: indicates whether the following field exists.
- `T_offset_upper_bound`: an upper bound of the time offset.
- `T_expire_time`: the expiration time of this tuple.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 5893 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Generation Message Time                    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                 Time Information for Peer #1                 |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
|                 Time Information for Peer #N                 |
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Format for Time Information:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 5893 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Peer  Address                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Last Received Time  (integer part)             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                Last Received Time  (fractional part)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Time Difference Lower Bound (integer part)          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Time Difference Lower Bound (fractional part)     |P|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 16: Format of the content of STP message, and of its Time Information sub-parts

6.1.4 STP message generation

Periodically, each node implementing OSTP generates STP messages as follows:

1. `Generation Message Time` is filled with the current time.
2. For each Time Protocol Tuple in the Time Protocol Information Set which is not expired, a "Time Information" field is added with:
   - `Peer Address` set to T_peer_address
   - `Last Received Time` set to the value of T_peer_last_time
   -
     - If T_offset_lower_bound_list has at least a couple which is not an expired value: `Time Difference` field is set to the minimum value of "time" of the non-expired couples in the T_offset_lower_bound_list.
     - Otherwise: `Time Difference` value is set to zero, and the `P` (present) bit is set to zero as well.

All expired time protocol tuples and all the expired couples in T_offset_lower_bound_list should also be removed.

6.1.5 STP message processing

Upon receiving a message from another node, the following steps are performed:

1. If no Time Protocol Tuple exists with T_peer_address == originator of the message:

   A new Time Protocol Tuple is created with:
   - T_peer_last_time = `Generation Message Time` of the message,
   - T_peer_address = originator address
   - T_expire_time = current time + TIME_PROTOCOL_HOLD_TIME
   - T_has_offset_upper_bound = false
2. Otherwise: T_expire_time of the existing Time Protocol Tuple is updated with: current time + TIME_PROTOCOL_HOLD_TIME
3. The processing continues with the previous, Time Protocol Tuple, existing or created.
4. If the field `Generation Message Time` of the message is lower then the field T_peer_last_time, then the processing of the message stops here. (Case: late STP message or replay).
5. The receiver checks whether the message includes "Time Information" field with its own address.
6. If the receiver address does not appear as an `address` in one Time Information field in the message: the processing of the message stops. (Reason: no proof of freshness).

7. Otherwise such a Time Information field exists, and the processing continues using the value:
   - ΔT = `Message Generation Time` - current time
8. The couple (ΔT, expireTime = current time + TIME_PROTOCOL_HOLD_TIME) is added to the list of T_offset_lower_bound_list. If the size of this list becomes larger than the parameter TIME_PROTOCOL_MAX_SAMPLE, the older entry is deleted appropriently.
9. If the `Time Difference Lower Bound` exists (as evidenced by the `P` "present" flag), in the Time Information field, the additional fields of the Time Protocol Tuple are set:
   - T_has_offset_upper_bound = true
   - T_offset_upper_bound = - `Time Difference Lower Bound`

6.1.6 STP message forwarding

STP messages are forwarded using MPR-flooding.

This is guaranted to function, because in a first step, the neighbors exchange clock information with each other; in a second step, they establish symetric links (with secure Hellos now passing the time-stamp check) and establish two hop neighbor information; in a third step, they compute proper MPRs; in a fourth step, STP messages can reach all nodes in the network through MPR-flooding via those MPRs; and in a fifth step, any secure OLSR message from any node is accepted by all other nodes.

6.1.7 Time-Stamp checking

The verification of time-stamps is modified when OSTP is used. When a time-stamp T_s is to be verified for an incoming message, the following processing is performed:

- If there exists no Time Protocol Tuple with T_peer_address == Originator address, then the time-stamp check is considered failed, and stops here.
- Otherwise there exists a Time Protocol Tuple.
- If this tuple has T_has_offset_upper_bound == false, then the time-stamp check is considered failed and stops here.
- If this tuple has no non-expired entries in the T_offset_lower_bound_list, then the time-stamp check is considered failed and stops here.
- Otherwise: Offset_min_lower_bound is computed as the minimum time value of the non-expired couples of that list.
- If T_offset_upper_bound < Offset_min_lower_bound, then the time-stamp check is considered failed and stops here (Case: temporary inconsistent clock).

- If T_offset_upper_bound - Offset_min_lower_bound, is greater then a given value TIME_PROTOCOL_MAX_RTT, similarly, the time-stamp check is considered failed, and the processing stops here.
- Otherwise the time-stamp check succeeds if and only if the time-stamp T_s is in the interval
[current time + Offset_min_lower_bound - TIMESTAMP_TOLERANCE, current time + T_offset_upper_bound + TIMESTAMP_TOLERANCE].

6.1.8 Proposed constant values

- `TIME_PROTOCOL_MESSAGE` = 0xd0
- `SIGNED_TIME_PROTOCOL_MESSAGE` = 0xd1
- `TIME_PROTOCOL_GENERATION_INTERVAL` = 60.0 seconds (and increased when the network becomes larger).
- `TIME_PROTOCOL_HOLD_TIME` = 600.0 seconds
- `TIME_PROTOCOL_MAX_SAMPLE` = 10
- `TIME_PROTOCOL_MAX_RTT` = 1.0 second
- `TIMESTAMP_TOLERANCE` = 15.0 seconds (may be lowered to `TIME_PROTOCOL_MAX_RTT`)

6.1.9 Validation

In this section, a given scenario is used to verify the proper functioning of the secured version of OLSR: the same scenario is run first without enabling OSTP (hence the nodes do not perform clock synchronization) and second, by enabling OSTP (hence the nodes exchange time-stamp protocol messages).

The same simulation as the one used for the validation of scenario 3 is used, see section 4.6.6.

In the scenario, 100 OLSR nodes are randomly placed on the area, the radio range is set to 0.15 (km), and the simulation is run for 300 seconds, without any motion.

Some results, without and with OSTP running respectively, are shown on figures 17 and 18: they show that with the time protocol, indeed, the nodes were able to acquire the offset of clocks of all the other nodes, and establish links and routes.

6.2 Modifying RFC 3626

6.2.1 Performing security checks

The position of the verification of security fields (signature, source interface address, timestamp) may be performed at different places according to desired optimization level.

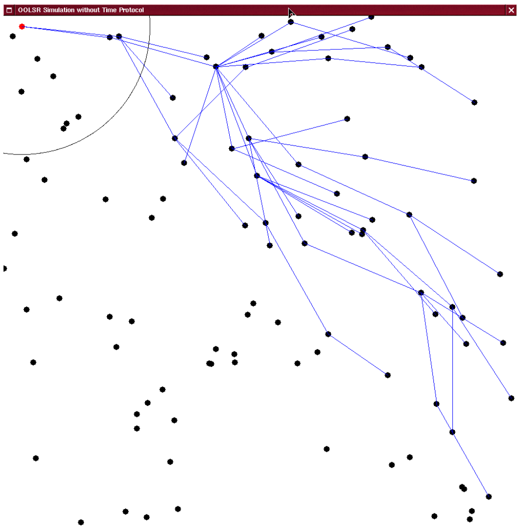

Figure 17: Simulation without OLSR Secure Time Protocol

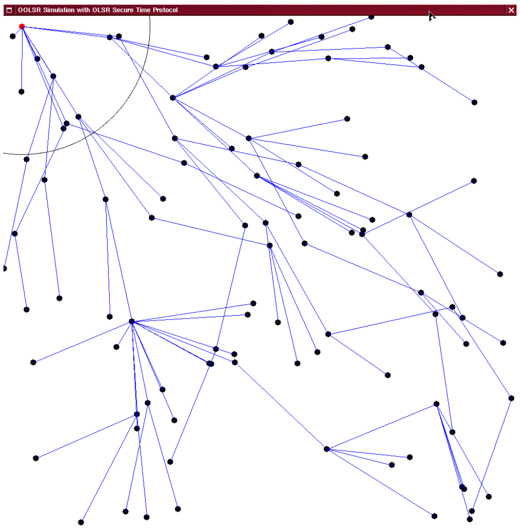

Figure 18: Simulation with OLSR Secure Time Protocol

The simplest way is to insert the tests in the steps of the algorithm given for "3.4 Packet Processing and Message Flooding", page 16, between steps 2 and 3:

```
2bis  If the message type is a SIGNED_HELLO_MESSAGE,
      SIGNED_TC_MESSAGE, SIGNED_MID_MESSAGE or SIGNED_HNA_MESSAGE,
      then:

      2bis.1 If the message has a format inconsistent with the
             signed message format (bad security information size,
             bad message content size, unsupported message type,
             unsupported authentication method), the message MUST
             silently be discarded.

      2bis.2 The signature of the message is verified
             on the sequence of bytes made from the whole message
             except the Signature field itself and with the Time
             To Live and Hop Count fields set to 0.
             If the signature is not verified, the message MUST
             silently be discarded.

      2bis.3 The time-stamp of the message is verified, by checking
             that ``time-stamp - current time'' is within
```

```
          TIMESTAMP_TOLERANCE. If there is no Time-Stamp field
          in the security information field or if the time-stamp
          is not within tolerance, the message MUST
          silently be discarded.

   2bis.4 If the message type is a SIGNED_HELLO_MESSAGE, the
          following check on the sender interface address is
          performed:

          2bis.4.1 If the sender interface address (NB: not
                   originator) is different from the "Source
                   Interface Address" in the security information
                   or if there is no such address is present:
                   the message MUST silently be discarded.
```

The previous modification has the drawback of verifying signatures of messages which may be neither processed nor forwarded ; an optimisation is to perform the checks of "2bis" only when they are to be used:

- In the step 3 "Processing condition", substep "3.2":

```
        3.2  Otherwise, if the node implements the Message Type of the
             message, the message MUST be processed according to the
             specifications for the message type,
             if and only if the security checks are passed.
```

- In the step "3.4.1. Default Forwarding algorithm", the step "4bis" should be added:

```
4bis  If the message type is a SIGNED_HELLO_MESSAGE,
      SIGNED_TC_MESSAGE, SIGNED_MID_MESSAGE or SIGNED_HNA_MESSAGE,
      the security checks are performed. If the security checks fail,
      the message MUST silently be discarded and its processing stops.
```

6.2.2 Signed message generation

Instead of generating Hello, TC, MID and HNA messages with the format given in RFC 3626, the node must generate such messages with the format given in section 3.2.2.

Depending on table 2, it must include time-stamp and source interface address when mandatory.

References

[1] T. Clausen (ed) and P. J. (ed), "Optimized Link State Routing protocol (OLSR)," October 2003, rFC 3626, Experimental.

[2] C. Adjih, T. Clausen, A. Laouiti, P. Muhlethaler, and D. Raffo, "OLSR security, theoretical study deliverabe 1 for CELAR also published as Securing the OLSR routing protocol with or without compromised nodes in the network'," Hipercom Project, INRIA Rocquencourt, Tech. Rep. INRIA RR-5494, February 2005.

[3] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-demand Distance Vector (AODV) routing," July 2003, rFC 3561, Experimental.

[4] D. B. Johnson, D. A. Maltz, and Y.-C. Hu, "The Dynamic Source Routing protocol for mobile ad hoc networks (DSR)," July 19 2004, internet-Draft, `draft-ietf-manet-dsr-10.txt`, work in progress.

[5] R. Ogier, F. Templin, and M. Lewis, "Topology dissemination Based on Reverse-Path Forwarding (TBRPF)," February 2004, rFC 3684, Experimental.

[6] C. Adjih, D. Raffo, and P. Muhlethaler, "Attacks against olsr: Distributed key management for security," in *2nd OLSR Interop / Workshop*, Palaiseau, France, July 28-29 2005.

[7] C. Adjih, T. Clausen, P. Jacquet, A. Laouiti, P. Muhlethaler, and D. Raffo, "Securing the OLSR protocol," in *Proceedings of the 2nd IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2003)*, Mahdia, Tunisia, June 25–27 2003.

[8] D. L. Mills, "Internet time synchronization: The network time protocol," in *Zhonghua Yang and T. Anthony Marsland (Eds.), Global States and Time in Distributed Systems, IEEE Computer Society Press*, 1994.

[9] ——, "Network time protocol (NTP)," Network Working Group Request for Comments: 958, pp. 1–14, 1985.

[10] ——, "On the accuracy and stability of clocks synchronized by the network time protocol in the internet system," *Computer Communication Review*, vol. 20, no. 1, pp. 65–75, 1990. [Online]. Available: citeseer.ist.psu.edu/mills90accuracy.html

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