Attacks Against OLSR: Distributed Key Management for Security

Cédric Adjih, Daniele Raffo, Paul Mühlethaler
INRIA, Domaine de Voluceau, France[1]

***Abstract*— In Mobile Ad Hoc Networks (MANETs), mobile nodes use wireless devices to create spontaneously a larger network, larger than radio range, in which communication with each other is made possible by the means of routing. One routing protocol for such MANET networks is OLSR, on which this article focuses. We examine the security issues, and describe an architecture including multiple securing mechanisms. The attacks prevented by this architecture, along with details about protocols, algorithms, mechanisms and implementation details are given.**

I. Introduction

Security in wireless ad-hoc networks is a requirement for many applications However, due to their wireless and distributed nature, ad-hoc networks have to rely on communication with (potentially out of sight) peers, and a number of attacks are possible.

This article is presenting a study of security issues related to integrity of an ad hoc network, and an architecture of security which is partly implemented. We only consider the use of the OLSR routing protocol [1]. Security of OLSR is not a new topic ; in previous works, studies of security issues in an OLSR Networks have been conducted [3]–[5], and several proposals for securing OLSR exist, including [3], [4], [6]–[8].

II. Overview

In this document, the starting point of the security architecture is the analysis of an attack tree of the routing protocol OLSR. The figure 1 represents such an attack tree. An

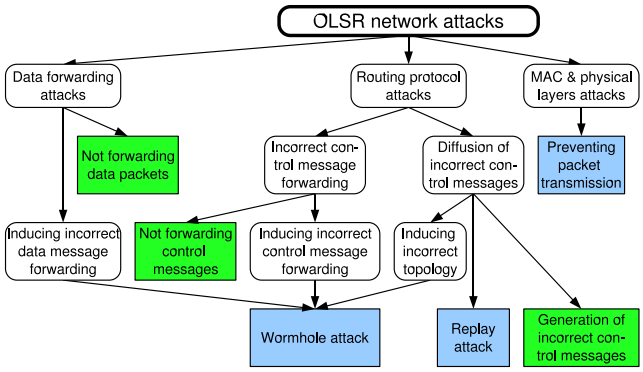

Fig. 1. OLSR Attack Tree (OR-tree)

attack tree is a formal way to investigate security (see [12]). Details of the attacks on the figure are found in [5] and [4]; confidentiality attacks (data and routing) are not considered; the colored squares correspond to the end attacks that should be prevented.

Our goal is to give a sketch of an overall balanced security solution, attempting to identify each attack, and searching for defenses. Many of the individual elements and mechanisms were already addressed in previous work

Before addressing attacks of figure 1, a remark is that in most security architectures, authentication is an essential and necessary component for avoiding threats. Our architecture is no exception, and indeed uses authentication, provided by cryptologic methods. Then network can then be divided in two parts: *authenticated* nodes, and *unauthenticated* nodes.

The main principles of the architecture are:

- For authenticated nodes: **trust but verify**. By default, the behavior of authenticated nodes is assumed correct. However it is assumed that one participant may start to act adversarily (in the following, an adversarial authenticated node is denoted *compromised node*), thus the policy is to perform ongoing checks.
- For unauthenticated nodes: **protection**. The aim is to prevent them to disrupt the network.

It is further described in this article ; the rest of the article is structured as follows: section III, presents the architecture built on the previous principles and uses a solution-oriented approach, which aims at preventing attacks from the tree on figure 1. In contrast, the complementary section IV, uses a problem-oriented approach, and gives a taxonomy with a detailed description of each problem (attack). The other sections describe components of the architecture: distributed key management in section V, study of cryptosystems in section VI, wormhole prevention in section VII, and timestamps in section VIII.

III. Architecture for Securing OLSR

A. Overview of Authentication Architecture

In the architecture, asymmetric (public key) cryptology is used. This is a requirement for the policy “trust but verify”, since with respect to authenticated nodes, a necessary complement of “verification” is the step the traceability and accountability: when a trusted node misbehaves, being able to identify it among other nodes, is a necessity and a deterrent.

With traditional asymmetric cryptology, an issue is that public keys need to be distributed, hence a PKI infrastructure is needed. Some efficient proposals and implementations already exist for OLSR, such as [8], where a distributed certificate

[1]This work was partly funded by DGA/CELAR.

authority is introduced in the network: threshold cryptography is used so that a node in the network only need to connect the closest k authorities (and allowing also server redundancy).

In the architecture, choice was made that some *Authentication Authority* would exist, but would not be necessarily present in the ad-hoc network. Hence a participant willing to join the network would first communicate with the Authentication Authority to exchange security parameters, such as public or private keys or certificates. Later it would enter in the network: with "identity-based" cryptography it would not need to diffuse its public keys ; however for performance reasons, choice was made that a node may use traditional asymmetric cryptography, and then an important point of the architecture is this distribution of the public key itself. The authentication architecture is further detailed in section V.

B. Preventing attacks from authenticated nodes

On fig. 1, the attacks specific from the authenticated nodes are the following: generation of incorrect control messages (see section IV-A), not forwarding data packets, not forwarding control messages (see section IV-B).

The prevention of incorrect control messages, was addressed in [10], by noting that since all OLSR information is about links, it is sufficient that both sides of a link sign an advertised link ; alternatively, in [2], each node originates link state messages, hence a remote node, can check whether consistent link state messages from both sides of any link were received. In both cases, one unique compromised node is unable to inject invalid topology in control messages. If several compromised nodes collaborate, they can inject artificial topology between each other, but not, however, between uncompromised nodes.

The other two attacks are compromised nodes that do not forward data or control messages (dropping them). Detecting this misbehavior is a difficult problem, and is the target of further work: a first step, by counting packets sent and received (at two hops distance), was introducted in [4]. It prevents one unique compromised node to drop packets ; ensuring that correct packets are received may be done as in section VII, and preventing attacks by several compromised nodes was researched in [14], [15].

C. Preventing attacks from unauthenticated nodes

With respect to unauthenticated nodes, the three attacks listed are: wormhole attacks (see section IV-B.3), replay attacks (see section IV-B.2) and preventing packet transmission.

About the attacks preventing packet transmission, at the physical and MAC level: defense against them is left as external methods; for instance jamming is typically addressed by the search of jamming devices.

The replay attack: it is the retransmission of some older control messages. It is prevented by the use of timestamps, see section VIII.

The wormhole attack is the one with the most potential consequences. A wormhole attack [5], [11] has the effect of creating artificially a link by repeating packets from one place of the network to another, inducing incorrect topology. This is effectively an attack when data packets, or control messages are dropped on this artificial link, entirely or selectively.

A defense for packet dropping is to use a variant of the counting techniques employed in [4]. Both ends of a link can discover that a message was sent but not received by techniques in sec. VII. If this occurs too frequently, any of the sides may decide that the *link is compromised*, and cease to advertise or use it. Another defense is to consider transmission and reception times based on precise clocks [9]: the difference should be close to the transmission delay.

Another attack which may be create by a wormhole attack, is a manufacturing of an impossible control message order, causing MPR flooding to drop them ("inducing incorrect message forwarding"), see section IV-B.4. This is a topic for further investigation.

D. Implementation of the Architecture

Part of the mechanisms presented previously have been, or are being implemented. The implementation is based on `OOLSR` [17]. An important parameter is that of course, recent equipment is targeted (e.g., with noticeable processing power); however one of our implementation targets is low-consumption, low-resource routers used in the demonstrator built at CELAR [16] (based on 486-133 Mhz microprocessors). This has implications.

The following table, reiterates the previously analyzed attacks and defenses, and indicates degree of implementation.

Attack	Defense
Unauthenticated nodes in the network	authentication, and key distribution, implemented
Not forwarding data packets	subject of further research with techniques from section VII and/or also [4], [14]
Not forwarding control messages	(same as above)
Generation of incorrect messages	see [10] (or the approach of [2])
Jamming	not addressed
Replay attack	timestamps, section VIII, implemented
Wormhole attack	section VII, to be implemented

IV. Attacks on the OLSR Protocol

This section gives further details and illustrations of attacks of the OLSR protocol (it originates from [3], see also [5]). While these vulnerabilities are specific to OLSR, they can be seen as instances of what other link state routing protocols, such as OSPF [35], are subject to.

A. Incorrect Control Message Generation

One way in which a node can misbehave is generating control messages which are incorrect according to OLSR.

1) Incorrect HELLO Message Generation: As an instance of *identity spoofing*, a misbehaving node X may send HELLO messages on behalf of node C (Figure 2). Subsequently, nodes A and B may announce reachability to C through their HELLO and TC messages. Furthermore, node X may choose

MPRs from among its neighbors, signaling this selection while pretending to have the identity of node C. Therefore, the chosen MPRs will advertise in their TC messages that they provide a "last hop" to C. Conflicting routes to node C, with possible connectivity loss, may result from this.

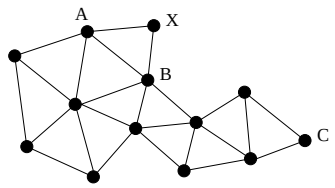

Fig. 2. Node X sends HELLO messages pretending to be C.

We call *link spoofing* the signalisation of an incorrect set of links in a control message. A misbehaving node X may perform link spoofing in its HELLO messages advertising a link with non-neighbor node A, as in Figure 3. One of the consequences is that C will have an incorrect MPR set, and messages originating from E and relayed through the MPR mechanism will not reach node A.

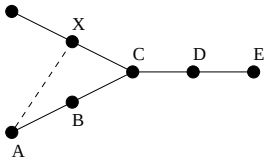

Fig. 3. Node X sends HELLOs advertising a fake link with A.

Node X can also misbehave by signaling an incomplete set of neighbors.

2) Incorrect TC Message Generation: TC messages with a spoofed originator address cause incorrect neighbor relationship to be advertised in the network. For instance, node X sends a TC message on behalf of node C, advertising A as a neighbor (Figure 4). Node D, upon reception of the TC message, will falsely conclude that C and A are neighbors. For this attack to be successful, the TC message must bear an `ANSN` (Advertised Neighbor Sequence Number) greater than the highest ANSN value referenced to C, as contained in any tuple of D's Topology Set; otherwise D will discard the TC message, according to the protocol.

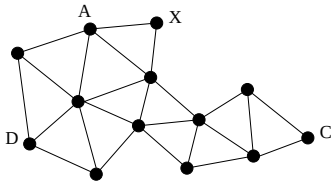

Fig. 4. Node X sends TC messages pretending to be C.

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

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

4) ANSN Attack: The misbehaving node may listen to a TC message from node A and record the ANSN of the message; then it sends a TC with a spoofed originator address of node A, and an ANSN much greater than the value recorded. According to the protocol specifications, nodes will ignore further TC messages from A as old information, because they bear a smaller ANSN. We call this an *ANSN attack*.

This attack can be spotted as the spoofed TC bears an ANSN much higher than that of the latest genuine TC message received from A, however the attack may be performed repeatedly.

B. Incorrect Control Message Relaying

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

1) Blackhole Attack: An attacker can drop received routing messages, instead of relaying them as the protocol requires, in order to reduce the quantity of routing information available to the other nodes. This is called *blackhole attack* by Hu et al. [32], and is a "passive" and a simple way to perform a Denial of Service.

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

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

2) Replay Attack: As topology changes, old control messages, while valid in the past, describe a topology configuration not existing anymore. An attacker can perform a *replay attack* by recording old valid control messages and resending them, to make other nodes update their routing tables with stale routes. This attack is successful even if control messages bear a non-timestamped digital signature.

A TC message cannot be replayed "as is" or it will not be accepted, because of the ANSN. Therefore the attacker needs to increase the ANSN of the TC message, causing indirectly an ANSN attack.

3) Wormhole Attack: The *wormhole attack* [33] is quite severe, and consists in recording traffic from one region of the network and replaying it "as is" in a different region. An extraneous $A-B$ link can be artificially created by an intruder node X by wormholing control messages between A and B (Figure 5). A longer wormhole can also be created by two colluding intruders X and X' (Figure 6). The created link is at the mercy of the attacker.

To successfully exploit the wormhole, the attacker must wait until A and B have exchanged sufficient HELLO

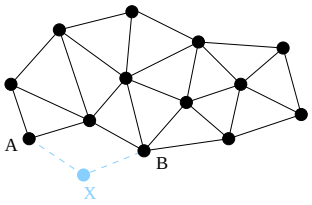

Fig. 5. A wormhole created by node X.

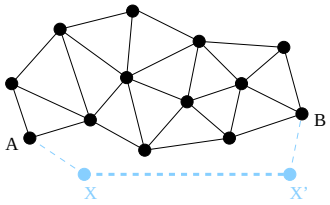

Fig. 6. A wormhole created by colluding nodes X and X'.

messages (through the wormhole) to establish a symmetric link. Until that moment, other tunneled control messages would be rejected, because the OLSR protocol specifies that TC/MID/HNA messages be not processed if the relayer node (the "last hop") is not a symmetric neighbor. However, once created, the $A - B$ link is at the mercy of the attacker.

4) MPR-Flooding Attack: The "first transmit rule", described in the OLSR specifications, states that a node receiving a message in MPR flooding checks if the sender is its MPR selector. If so, the node retransmits the message. If the sender is not a MPR selector of the node, the latter will never retransmit anymore the message.

It could be exploited to impede the correct relaying of control messages and we call the related misbehavior a *MPR attack*. Consider the following scenario (Figure 7): node A sends a message to its neighbors B and X, where B is MPR of A, and X is not a MPR. The misbehaving node X does not select properly its MPR set, and retransmits the message (even if it is not supposed to) which is received by C. Node B retransmits the message to C, where C is MPR of B. The crucial point is that C, even being a MPR, will not relay the message because C has already received it from X.

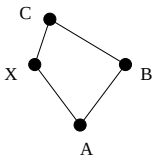

Fig. 7. Node X performs a MPR attack.

C. Gaining Priviledged Position

A complement to other attacks is the fact that a node can force its election as MPR by setting the `Willingness` field to the WILL_ALWAYS constant in its HELLOs. According to the protocol, its neighbors will always select it as an MPR. Using this mechanism, a compromised node can easily gain, as an MPR of some nodes, a priviledged position inside the network and later exploit it. In this case, a countermeasure is to allow each node to select two (or even more) MPRs to cover all its 2-hop neighborhood (at the expense of MPR flooding efficiency). This remedy can also be used to prevent incorrect MPR selection, which is an effect of some of the attacks explained in the previous sections.

V. Distributed Key Management

Our architecture relies on asymmetric cryptographic techniques. In this section, this general architecture is detailed, while details about cryptography are in section VI. The architecture relies on two levels of key distribution:

- **PKI Interaction:** an (offline) PKI Authority is in charge of certifying or assigning keys of each node participating in the trusted network. Each node joining the network will have the public key of of the certification authority, as pictured on figure 8. This key is denoted the *global key*.
- **Key Distribution:** later, any node entering the ad-hoc network could diffuse its public keys, with a specific key exchange protocol, as pictured on figure 9, with proper parameters, certificates and signatures. The key which is used later to sign message is denoted the *local key*, and can be either its global key, or newly generated private/public keys.
- **Protocol Message Signing:** at the same time, the node would start originating OLSR control messages, signing them using the local key with a specific extension (see figure 10) which prepends a special signature message.

However, the architecture targets two kinds of different asymmetric cryptography: identity-based techniques, where the public key of a node is derived from the some identifier of the node (see sec. VI), other traditional asymmetric techniques, where a public key exists but must be distributed (possibly with certificate).

The previous presentation and figures were accurate in the case of traditional asymmetric techniques. But when identity-based systems are used, the PKI Authority doesn't need to issue a certificate for the key of the node: in essence, the key is "self-certifying". There are two levels of signatures (global and local keys), each of which can use either technique. The following table details the key distribution method, depending on the choice at each level:

Global key	Local key	Key distribution
identity-based	same	not needed
identity-based	traditional asymmetric	distribution of local key
identity-based	different, identity-based	node can act as a IBS key server for other nodes, and distributes its local public master key with all IBS parameters
traditional asymmetric	identity-based	same as above, but with a certificate for global key
traditional asymmetric	same/different; traditional asymmetric	distribution of local public key and certificate for global key

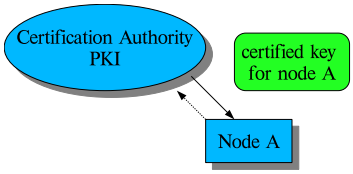

Fig. 8. Authentication Authority

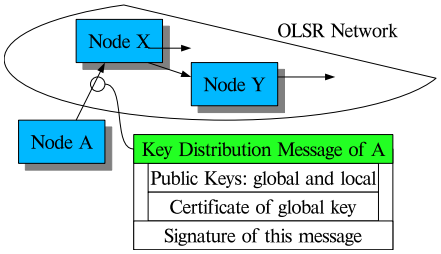

Fig. 9. Key Distribution

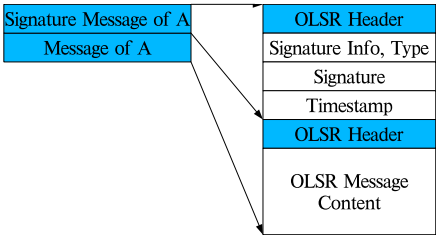

Fig. 10. Signing

As represented on figure 9 and in the table, a special protocol operates in order to exchange keys. In the case of an global key, with traditional asymmetric cryptology, the node originates *Key Distribution Messages* which include: the global public key, a certificate of the global public key which includes the node identifier (originator IP address), and, when the local key is different from the global key, all the parameters of the local key. To authenticate the message, it is signed with the global key itself. In the currently implemented protocol, a pure flooding of this message is performed.

When the global key is identity-based, no certificate is necessary, and only information about the local key (if any) is transmitted (with a signature).

VI. Cryptosystems

Our security architecture relies on the use of asymmetric keys. Several such cryptosystems exist, and a good reference for mature and state-of-the-art cryptosystems is the IEEE P1363 standardisation group for public-key cryptography [31].

Asymmetric key systems include:

- Pairing-based systems
- Identity-based systems (some of which are pairing-based)
- Traditional public key cryptography

Each system offer different features. Some pairing-based systems such as BLS signatures [18] offer short signatures with many cryptographic extensions (blind, aggregate, threshold signatures, multisignatures, batch-verification with [24], ...; see [19] for a review of pairing-based systems).

Some of these pairing-systems, are "identity-based" systems. The principle of identity-based systems is simple: the public key of a signing entity is derived from an identifier of the entity (here the IP address). Naturally, the corresponding private key is generated by the Authentication Authority. Identity-based systems remove the need to transmit a certificate, or have a certifying authority present in the network.

Traditional public key cryptography offers systems based on integer factorization or discrete logarithm (faster than previous methods), and elliptic curve discrete logarithm (shorter signatures).

A. Implementation and empirical evaluation of cryptosystems

In our current implementation of the security architecture, two different pairing-based systems are used for signatures:

- Identity-based signatures with Cha-Cheon signatures [20] (pairing-based), for the *global keys*.
- BLS short signatures for the *local keys*.

Both are prototypes designed to investigate cryptosystems, and the global keys have realistic size but implementation is slow, while local keys have irrealistic sizes but a faster implementation.

For both cases, a Weil pairing is used on supersingular elliptic curves of embedding degree $k = 1$, using the family of such curves proposed in [21], and with the following parameters:

Key	Method	Elliptic curve and parameters	Key (Size)	Security
Global Key	CC [20]	$n_g = 2^{174} + 2^{115} + 1$, $p_g = (2^{339} n_g)^2 + 1$, $E : y^2 = x^3 - x$	$(U, V) \in E(\mathbb{F}_{p_g})^2$	$\sim$ 80 bits, $\sim$ RSA1024
Local Key	BLS [18]	$n_l = 2^{14} + 2^5 + 1$, $p_l = (2n_l)^2 + 1$, $E : y^2 = x^3 - 4x$	$s \in E(\mathbb{F}_{p_l})$	none in practice

According to [21], elliptic curves of degree 1 are an possible but uncommon choice. Then, many optimisations for other degrees > 1 are not available, and the resulting signature key size is larger for an identical security level. However as $k = 1$, there is no need to consider the elliptic curve on an algebraic extension of the field $\mathbb{F}_p$; hence, implementation is simpler.

The obtained results are highlighted in the table which follows. For our proof of concept implementation for local key signature with 32 bits integers (hence no security), most of the time is spent in the computation of an hash of the message, but still, considering the pairing timings, the results obtained already show that the computational power of the 486 (see sec. III-D) is too small to have a chance of implementing real pairings.

Key system	486 (133Mhz)	PIII (1 GHz)	P4 (2.8 Ghz)
Global Key Sign. (CC)	18.3 s	0.51 s	0.25 s
Global Key Verif. (CC)	77.3 s	2.12 s	1.08 s
Local Key Sign. (BLS)	30 ms	1.1 ms	0.48 ms
Local Key Verif. (BLS)	43 ms	1.57 ms	0.72 ms
Local Key, Weil pairing	7.83 ms	0.28 ms	0.12 ms

In reality, although there exists identity-based systems with signature methods which are noticeably more efficient [22], we see that pairing computation is already a bottleneck. Benchmarking other systems, which are considerably more mature, we found $150ms$ for a signature for a PIII 1 Ghz with [23]. For current security levels, Tate pairing can give faster results: $32.5ms$ in [27]; $20ms$ for a 512-bit Tate pairing with [26], also in [28], with $53ms$ for one BLS verification.

Traditional asymmetric cryptography is more suitable for 486-133 Mhz. Using `crypto++` [25], results are: ESIGN-1023 sign/verif= 30ms/12ms; RSA-1024 = 380ms/12ms; RW-1024=370ms/6ms; LUC=1024=590ms/15ms. Note that signatures longer than verification, match perfectly OLSR networks.

VII. Wormhole Attack Prevention

In order to prevent wormhole attacks, a variant of the counting technique [4] is used. With such techniques, each node periodically declares how many messages it had received from neighbors. If a compromised link drops packets, a counting mismatch will be detected.

However, to prevent compromised links to mutate packets (instead of just dropping them), instead of a simple count, each node advertises a set of hashes of of the packets received over each of the last k intervals, using [13], allowing neighbors to check whether packet loss crosses a threshold. If it is the case, the link is assumed compromised and is no longer advertised.

VIII. Timestamp

In our architecture, replay attacks are prevented by the traditional use of timestamps. A message is rejected if the timestamp is judged to be too old. The timestamp used here is based on wall clock. There are two possibilities: synchronized or unsynchronized clocks.

The main issue is that with unsynchronized clocks, each OLSR node should first determine its clock difference with any other node in the network in a secure way. This requires typically a N^2 messages exchanges, associated with a challenge/response protocol. In [6], the standard challenge/response was optimized to limit emissions to periodic N message exchanges (each transmitted to the whole network), while in [7], the challenge/response protocol was run once but for each pair of nodes.

Because of the complexities of unsynchronized clocks, in this current approach, clock synchronization is assumed. Hence the functioning is the following:

- The clock of one node has an initial synchronization, with some authentication authority, when key information is exchanged.
- The clock of one node is synchronized (by an unspecified mechanism) from time to time (for instance manually) or direct connection to a trusted clock reference.

A timestamp is rejected if it is too far from the current time, e.g. $|Timestamp - t| \geq \Delta t$. A logical choice for Δt is to set it close to the holding time of duplicate messages. Indeed, once a message is in the duplicate table (at first reception), further duplicate transmissions will be ignored as long as the duplicate tuple exists, protecting against replay attacks.

Then, the main issue when synchronizing clock becomes the precision of the clocks. In most computers and network equipment, piezo-electric quartz oscillators are used to keep track of time (in PCs, it is the so-called *BIOS clock*), and the precision of those clocks is limited ; in absolute terms, on the order of magnitude of one second per day. For instance, [29] reports a maximum clock drift between nodes was 0.7 seconds per day on a cluster of 8 PCs. This drift is due to two factors: the lack of precision of the quartz (assumed to oscillate at a certain frequency F_0, different from the actual one F_r), and also the variations in frequency F_r due to temperature, aging, vibration-induced noise, ... [30]. The first factor dominates.

A. Empirical study of time synchronization

In order to assess the frequency needed for resynchronization, experiments were conducted with the routers targetted by the OLSR implementation (see sec. III-D): 4 routers were broadcasting their BIOS clock periodically on an Ethernet. A machine recorded the arrival times of the different broadcast of the clocks, along with their advertised time value.

First the time drift is presented on figure 11: one of the machines, `router 1` is used at reference, and the difference between the value of clocks of the nodes is plotted as a function of the experiment. Clocks are synchronized at the begining of the experiment. As one can see, the maximum clock drift (between `router 2` and `router 3`, is around **1 second per day**), quite comparable with expected values.

However the second observation is that the curves of figure 11 are mostly linear, confirming that the main factor in clock drift is incorrect calibration of F_r. Using linear regression on the values found, the linear component of drift between one router and the `router 1` was removed. In practice such corrections could be performed by making precise time difference measurements at two separated points of time. The figure 12 shows the results. The first observation, is that the precision is much better, the drift is around **30 ms per day**. However the shape of the curves is not convincing, for instance the curve of the router `router 2` is noticeably irregular. Comparing the drift estimates based 1) on measurements from day=4 to day=6, to 2) drift estimates based on measurements from day=7.5 to day=9.5 (worst case discrepancy), the discrepancy between the estimates is found to be about **0.2 second per day**. Hence it is one order of magnitude more that before, but nearly one order of magnitude than without linear drift correction.

The following table gives a summary of the equivalent necessary synchronization intervals, for a drift of maximum **15 seconds**, based on the measurement results:

Method	unique synchronization	drift correction (average)	drift correction (worst)
Measured drift	1 sec per day	30 ms per day	0.2 sec per day
Needed synchronization interval	every 15 days	every 500 days	every 75 days

IX. Conclusion

This article presented issues of OLSR security, reviewed some of the existing litterature addressing them, and proposed an architecture to secure OLSR, which is being implemented.

Acknowledgments

The authors want to thank François Morain and Andreas Enge, for information and pointers about cryptography, including identity-based systems.

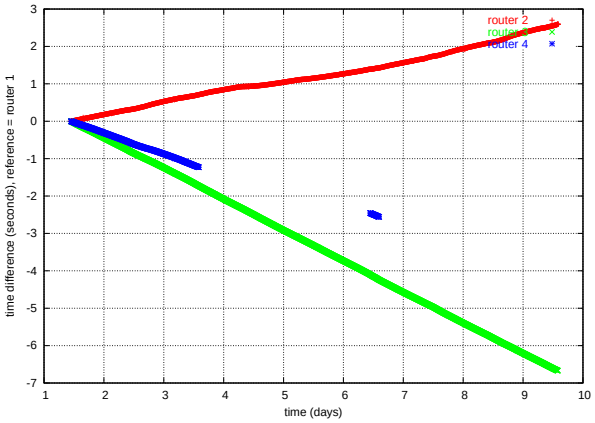

Fig. 11. Time difference between nodes

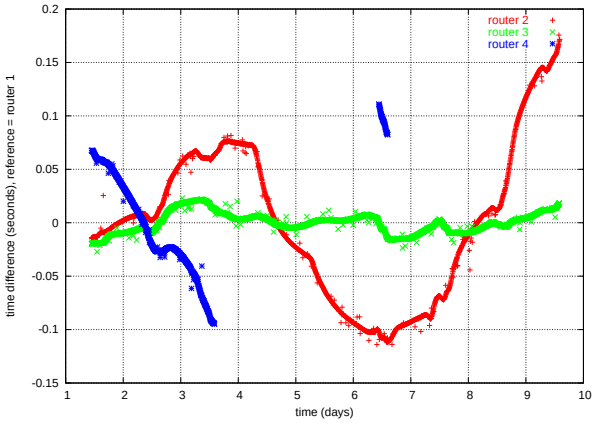

Fig. 12. Time difference when synchronising clocks

REFERENCES

[1] Clausen, Thomas Ed. - Jacquet, Philippe Ed. - Adjih, Cédric - Laouiti, Anis - Minet, Pascale - Mühlethaler, Paul - Qayyum, Amir - Viennot Laurent *The Optimised Routing Protocol for Mobile Ad-hoc Networks: protocol specifi cation*, INRIA Research Report RR-5145, March 2004 ; also "*Optimized Link State Routing Protocol*", IETF: The Internet Engineering Task Force, RFC 3626, October 2003

[2] R. Perlman, *"Network Layer Protocols with Byzantine Robustness"*, Ph.D. thesis, Massachusetts Institute of Technology, October 1988.

[3] Daniele Raffo, *"Security Schemes for the OLSR Protocol for Ad Hoc Networks."*, Ph.D. thesis, Université Paris 6, to be defended in September 2005.

[4] Cédric Adjih, Thomas Clausen, Anis Laouiti, Paul Mühlethaler, and Daniele Raffo. *"Securing the OLSR Routing Protocol With or Without Compromised Nodes in the Network"*. Technical Report INRIA RR-5494. HIPERCOM project, INRIA Rocquencourt, February 2005.

[5] Thomas Clausen (ed) and Emmanuel Baccelli (ed). *"Securing OLSR Problem Statement"*. Internet-Draft, draft-clausen-manet-solsr-ps-00.txt, work in progress. IETF MANET Working Group, February 14 2005

[6] Cédric Adjih, Thomas Clausen, Philippe Jacquet, Anis Laouiti, Paul Mühlethaler, and Daniele Raffo. *"Securing the OLSR Protocol"*. In Proceedings of the 2nd IFIP Med-Hoc-Net 2003, Mahdia, Tunisia, June 25-27 2003.

[7] A. Hafslund, A. Tnnesen, J. Andersson, R. Rotvik, Kure, *"Secure Extension to OLSR"*, OLSR Interop Workshop, 2004.

[8] D. Dhillon, T. Randhawa, M. Wang, L. Lamont, *"Implementing a Fully Distributed Certifi cate Authority in an OLSR MANET"*, WCNC 2004 IEEE, Atlanta, Georgia, USA

[9] Daniele Raffo, Cdric Adjih, Thomas Clausen, and Paul Mhlethaler. *"Securing OLSR Using Node Locations"*. In Proceedings of 2005 European Wireless (EW 2005), Nicosia, Cyprus, April 10-13 2005.

[10] 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), Washington DC, USA, October 25 2004.

[11] Kimaya Sanzgiri, Bridget Dahill, Brian N. Levine, Clay Shields, and Elizabeth M. Belding-Royer. *"A Secure Routing Protocol for Ad hoc Networks"*, ICNP November 2002.

[12] Bruce Schneier *"Attack Trees – Modeling security threats"* Dr. Dobb's Journal December 1999

[13] Burton H. Bloom, *"Space/time trade-offs in hash coding with allowable errors"*, Communications of the ACM, Volume 13, Issue 7 (July 1970), pp 422-426.

[14] I.Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy, *"Highly secure and effi cient routing"*, in Proc. IEEE Infocom 2004, Hong Kong, Mar. 2004.

[15] I.Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy, *"Amendment to: Highly secure and effi cient routing"*, Feb. 2004.

[16] T. Plesse, J. Lecomte, C. Adjih, M. Badel, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, A. Plakoo. *"OLSR performance measurement in a military mobile ad-hoc network"*, Int. Workshop on Wireless Ad-Hoc Networking, Tokyo, Japan, March, 2004.

[17] OOLSR, implementation of the OLSR protocol, INRIA, `http://hipercom.inria.fr/OOLSR/`, 2003-2005

[18] D. Boneh, B. Lynn, H. Shacham, *"Short signatures from the Weil pairing,"* Advances in Cryptology, Asiacrypt'2001, Lecture Notes on Computer Science 2248 (2002).

[19] R. Dutta, R. Barua, P. Sarkar, *"Pairing-Based Cryptography : A Survey"*, Cryptology ePrint Archive, Report 2004/064.

[20] J. C. Cha, J. H. Cheon, *"An Identity-Based Signature from Gap Diffi e-Hellman Groups"*, Practice and Theory in Public Key Cryptography PKC'2003, Lecture Notes on Computer Science 2567.

[21] N. Koblitz, A. Menezes, *"Pairing-based cryptography at high security levels"*, technical report, CACR 2005-08, Centre for Applied Cryptographic Research, University of Waterloo, 2005.

[22] Paulo Barreto, private communication.

[23] Security Lab in the Computer Science Department at Stanford University, `ibe-0.7.2.tgz` implementation of Identity-Based Encryption (and signature), `http://crypto.stanford.edu/ibe/`.

[24] J.H Cheon, Y.D. Kim, H.J. Yoon *"A New ID-based Signature with Batch-Verifi cation"*, Cryptology ePrint Archive, Report 2004/131.

[25] W. Dai et al., *"Crypto++"*, `http://www.eskimo.com/~weidai/cryptlib.html`

[26] M. Scott, `MIRACL`, *Multiprecision Integer and Rational Arithmetic C/C++ Library*, `http://indigo.ie/~mscott/`

[27] S. Galbraith, K. Harrison, D. Soldera, *"Implementing the Tate pairing"*, HP Laboratories Research Report HPL-2002-23, March 2002.

[28] P.S.L.M. Barreto, H.Y. Kim, B. Lynn, M. Scott, *"Effi cient Algorithms for Pairing-Based Cryptosystems"*, Crypty'2000, LNCS 2442, pp 354–368.

[29] Jorji Nonaka, Gerson H. Pfi tscher, Katsumi Onisi, Hideo Nakano *"An Evaluation of Low Cost Hardware-assisted Internal Clock Synchronization in PC Cluster Environment"*, PDPTA 2002, pp 456-461.

[30] J. R. Vig, *"Introduction to Quartz Frequency Standards"*, SLCETTR - 92-1 (rev. 1), Army Research Laboratory, Electronic and Power Sources Directorate, Fort Monmouth, NJ, October 1992.

[31] The IEEE P1363 Working Group, Standard Specifi cations For Public-Key Cryptography.

[32] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the 8th ACM International Conference on Mobile Computing and Networking*, September 2002.

[33] 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.

[34] Leslie Lamport, Robert Shostak, and Marshall Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.

[35] J. Moy. OSPF version 2, April 1998. RFC 2328, Standards Track.