An Advanced Signature System for OLSR

Daniele Raffo
INRIA Rocquencourt
Daniele.Raffo@inria.fr

Cédric Adjih
INRIA Rocquencourt
Cedric.Adjih@inria.fr

Thomas Clausen
INRIA Rocquencourt
T.Clausen@computer.org

Paul Mühlethaler
INRIA Rocquencourt
Paul.Muhlethaler@inria.fr

ABSTRACT

In this paper we investigate security issues related to the Optimized Link State Routing Protocol – one example of a proactive routing protocol for MANETs. We inventory the possible attacks against the integrity of the OLSR network routing infrastructure, and present a technique for securing the network. In particular, assuming that a mechanism for routing message authentication (digital signatures) has been deployed, we concentrate on the problem where otherwise "trusted" nodes have been compromised by attackers, which could then inject false (however correctly signed) routing messages. Our main approach is based on authentication checks of information injected into the network, and reuse of this information by a node to prove its link state at a later time. We finally synthetize the overhead and the remaining vulnerabilities of the proposed solution.

Categories and Subject Descriptors: C.2.2 [Network Protocols]: Routing Protocols

General Terms: Security.

Keywords: OLSR, ADVSIG, multiple signatures, proofs, link state, proactive.

1. INTRODUCTION

A Mobile Ad hoc NETwork (MANET) is a collection of nodes which are able to connect on a wireless medium to form an arbitrary and dynamic network. Implicit herein is the characteristic of the network topology to change over time as links in the network appear and disappear.

In order to enable communication between any two nodes, a routing protocol is employed. The abstract task of the routing protocol is to discover the topology (and, as the the network is dynamic, continuing changes to the topology) to ensure that each node is able to acquire a recent image of the network topology to construct routes.

Currently, two complementary classes of routing protocols exist in the MANET world. Reactive protocols (such as AODV [18] and DSR [11]) acquire routes on demand, while proactive protocols (such as OLSR [5], OSPF [14], DSDV [19], and TBRPF [17]) ensure that topological information is maintained through periodic message exchange.

SASN'04, October 25, 2004, Washington, DC, USA.

1.1 Security Issues: Related Work

A significant issue in the ad hoc networking domain is that of the integrity of the network itself. Routing protocols allow, according to their specifications, any node to participate in the network, with the assumption that all nodes are trusted and follow the protocol. If the network is subject to malicious nodes, the integrity of the network fails.

The primary issue with respect to securing MANET routing protocols is thus that of ensuring network integrity, even in the presence of malicious nodes. Security extensions to the reactive protocols AODV and DSR exist, respectively in the form of SAODV [24] and Ariadne [7]. SAODV uses digital signatures on Route Request and Route Reply messages. Ariadne authenticates the sender by using clock synchronization and delayed key disclosure. Another reactive protocol, ARAN [23], uses an authenticated route discovery. As concerns the proactive protocols OLSR and OSPF, a system of digital signatures has been proposed in [1, 20, 15]. The secured version of DSDV, named SEAD [6], uses hash chains for message authentication.

Maintaining the integrity of the network becomes more difficult when an intruder has compromised a trusted node (which hence becomes a malicious node) or has captured its private key; the intruder then becomes able to send authenticated messages. Known security techniques against this kind of attack which aim at identifying and blacklisting the faulty nodes, are the Watchdog/Pathrater [13], CONFIDANT [3] and WATCHERS [2, 9].

In this paper we will investigate security issues in the OLSR protocol, providing an improved security extension to ensure network integrity and avoid nodes misbehavior.

1.2 Paper outline

The rest of this paper is organized as follows: section 2 presents an overview of the OLSR protocol. Section 3 describes the vulnerabilities of OLSR, giving an example of the threats to which any proactive adhoc routing protocols is vulnerable. Section 4 provides a detailed description of our proposed security solution, as well as evaluates the overhead it incurs. The paper concludes with section 5.

2. THE OLSR PROTOCOL

The Optimized Link State Routing protocol (OLSR) [5, 10, 4] is a proactive link state routing protocol for mobile ad hoc networks. OLSR employs an optimized flooding mechanism to diffuse partial link state information to all nodes in the network.

2.1 OLSR Control Traffic

Control traffic in OLSR is exchanged through two different types of messages: HELLO and TC messages.

HELLO messages are emitted periodically by a node and contain a list of neighbors from which control traffic has been heard, a list of neighbors with which bidirectional communication has been established, and a list of neighbors that have been selected to act as a Multipoint Relay for the originator of the HELLO message – see section 2.2.

Upon receiving a HELLO message, a node examines the lists of addresses. If its own address is included in the addresses encoded in the HELLO message, bi-directional communication is possible between the originator and the recipient of the HELLO message.

In addition to information about neighbor nodes, periodic exchange of HELLO messages allows each node to maintain information describing the links between its neighbor nodes and nodes which are two hops away. This information is recorded in a nodes 2-hop neighbor set and is utilized for MPR optimization. Here we mention some constants that are used as link state values in HELLO messages, and which we will use when explaining our secured protocol: ASYM_LINK denotes an asymmetrical link, SYM_LINK denotes a symmetrical link, SYM_NEIGH denotes that the node is a symmetric neighbor, and MPR_NEIGH denotes that the node has been selected as MPR by the sender.

TC messages, just like HELLO messages, are emitted periodically. The purpose of a TC message is to diffuse link state information to the entire network. Thus, a TC message contains a set of bi-directional links between a node and some of its neighbors.

TC messages are flooded to the entire network, exploiting the MPR optimization described in section 2.2. Only nodes which have been selected as an MPR generate TC messages.

An individual OLSR control message can be uniquely identified by its `Originator Address` and `Message Sequence Number` – both from the message header. This will become important when discussing message signatures.

2.2 Multipoint Relay Selection

The core optimization in OLSR is that of *Multipoint Relay* (MPR) nodes. Each node must select MPRs from among its neighbor nodes such that a message emitted by it and repeated by its MPRs will be received by all nodes two hops away. MPR selection is performed [21] based on the 2-hop neighbor set received through the exchange of HELLO messages, and is signaled through the same mechanism.

Each node maintains a *MPR selector set*, containing the set of neighbors which have selected it as an MPR. This information is advertised in its TC messages.

Figure 1 shows a node with neighbors and 2-hop neighbors. In order to achieve a network-wide broadcast, a broadcast transmission needs only be repeated by just a subset of the neighbors. This subset is the *MPR set* of the node.

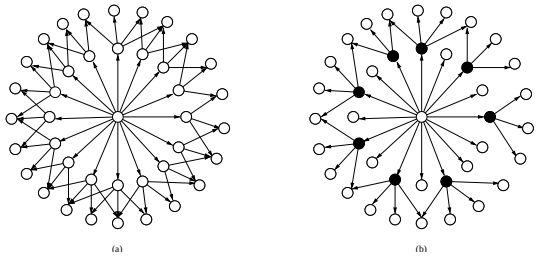

Figure 1: The broadcast from the central node is retransmitted: (a) by all its neighbors (b) by its MPRs only (solid circles)

3. VULNERABILITIES

In this section we discuss various security risks in OLSR. While these vulnerabilities are specific to OLSR, they can be seen as instances of what all proactive routing protocols are subject to.

Under a proactive routing protocol, each node must correctly generate routing control traffic conforming to the specification, and forward routing control traffic on behalf of other nodes. By carrying out an attack against the routing protocol, an intruder can perturb or paralyze the whole network. Often, the intruder will first need to gain full control of a trusted node, which then will start misbehaving. In the rest of this section we will show how routing misbehavior may appear in OLSR.

3.1 Incorrect Traffic Generation

One of the primary attacks against a routing protocol is to have nodes that generate incorrect control traffic, i.e. control traffic which appears to be valid but which in fact does not contain correct information. In this section we describe the various forms of incorrect traffic generation which an OLSR network may be subject to.

3.1.1 Identity spoofing

Identity spoofing implies that a misbehaving node sends control messages while pretending to be another node. Node X sends HELLO messages, with the originator address set to that of node A, as illustrated in figure 2. This may result in the network containing conflicting routes to node A. Specifically, node X will choose MPRs from among its neighbors, signaling this selection while pretending to have the identity of node A. The MPRs will, subsequently, advertise in their TC messages that they can provide a "last hop" to node A. Conflicting routes to node A, with possible loops, may result from this. Similarly, TC messages with a spoofed originator address cause incorrect links to be advertised in the network.

3.1.2 Link spoofing

Link spoofing implies that a node sends control messages signaling an incorrect set of neighbors. A misbehaving node advertising in its HELLO messages a neighbor relationship to non-neighbor nodes may cause inaccurate MPR selection, with the result that some nodes may not be reachable in the network. Again, TC messages which include non-existing links may result in routing loops and conflicting routes in the network (figure 2).

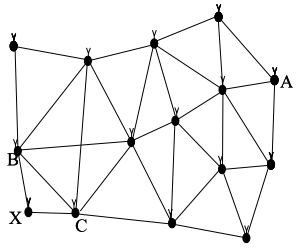

Figure 2: Node X sends HELLOs with the spoofed identity of node A; as a consequence, nodes B and C may mistakenly announce reachability to A through their TCs. Node X may also generate incorrect TCs advertising a spoofed link with A, causing other nodes to store a wrong topology of the network.

A node may also misbehave by signaling an incomplete set of neighbors, which might therefore lead to a breakdown in connectivity with the rest of the network. This purports to issuing incomplete, rather than false, information, and is not discussed in this paper. Furthermore, *denial of service* attacks, included those against the physical layer (jamming, radio interference, etc.), are not handled either.

While identity spoofing can easily be thwarted by the use of private keys for message authentication, withstanding against link spoofing is far more tricky. The security system we propose in this paper is effective against both attacks.

3.2 Incorrect Traffic Relaying

Another primary attack against an OLSR network is when a node in the network does, somehow, not relay traffic as is expected. This may occur in two different ways, detailed in this section.

3.2.1 Failure in relaying

If TC messages are not properly relayed the network may experience a breakdown in connectivity, i.e. some nodes may be unreachable.

3.2.2 Wormhole attack

A wormhole attack [8] is a severe attack in which traffic from one region of the network is recorded and replayed in a different region. As regards OLSR, an attacker may use an intruder node which is in the neighborhood of both A and B to relay HELLO messages from A to B and vice versa. An attacker could also use two colluding intruder nodes, one in the neighborhood of A and the other in the neighborhood of a distant node Z, connected via a direct wireless or wired carrier; the attacker may then tunnel HELLO messages through this longer carrier to create an extraneous A–Z link. In the OLSR protocol, where links are discovered by testing reception, this will result in extraneous link creation and therefore nodes storing an incorrect topology of the network.

Our security system is still vulnerable to the wormhole attack, and to failure in relaying.

4. OLSR WITH ADVSIG

To thwart attacks coming from compromised nodes, we introduce an advanced signature technique for the OLSR protocol. As in [1], it relies on creating and sending an additional message in conjunction with routing control messages. In this technique, this additional message now carries multiple signatures from different nodes. We call such a message an ADVSIG (for ADVanced SIGnature) message. This solution does not require modification of the standard OLSR control messages.

A PKI and a timestamp synchronization algorithm between all nodes of the network is required. For brevity, we do not address such issues in this paper; we refer implicitly to the infrastructure deployed in [1].

4.1 Introduction: Main Ideas

In OLSR, and in any other link state protocol, the network topology, with respect to the local neighborhood of a node, is related to what the network topology was at a previous instant. This because the link state at a given time T depends on the link state at an immediately previous time $T - \delta t$. E.g. node A declares a symmetrical link with node B, at time T. We can therefore state that at time $T - \delta t$ node B declared it had an asymmetrical link with A (i.e. B hears A), and declared this fact in a HELLO message which was received by A. In fact, this is exactly the way the nodes verify and establish symmetrical links in order to build a connected network [5]. Here we assume that all nodes correctly follow the protocol.

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

4.2 Link Atomic Information

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

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

1. the address of B
2. B's link state with respect to A
3. the timestamp of the creation time
4. the signature of these three fields, signed by A

The "address" and "link state" fields are exchanged through a HELLO message, respectively in `Neighbor Interface Address` and `Link Code`. The "timestamp" and "signature" fields will be contained in the ADVSIG message coupled to that

HELLO. Depending on its use, this atomic information is called either a *Certificate* or a *Proof*.

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

Later, when the node sends a HELLO or TC message, it will select the relevant Proof from its Certiproof Table and include it in the ADVSIG message coupled to that HELLO/TC message. (For some values of link state, as for asymmetrical links where only one node hears the neighbor, it is not possible to give a Proof: see section 4.3.)

Note that we call the same atomic piece of information a *Certificate* when it is created and supplied to inform about the neighborhood; and we call it a *Proof* when it is reused and supplied to prove a link state. Therefore, the Certiproof table of node B contains Certificates signed by various neighbors of B; in each of these Certificates, the "address" field contains the address of B.

4.3 Required proofs

As mentioned above, if node A wishes to report a link (in a HELLO/TC message) with a neighbor node B, the required proof must be built using elements of a HELLO message and the accompanying ADVSIG message that were recently sent by node B. The proofs are then stored (as Certificates) in the Certiproof table and reused (as Proofs) whenever necessary. The proofs must be sent along packed in a new companion ADVSIG message, with the new HELLO/TC messages they are intended to prove.

We give below a scheme of the required proofs. This scheme is based on the OLSR specifications [5], but see section 2.1 for a brief explanation.

When A wishes to report a link of type λ with B in a HELLO, the proof must be a recent HELLO from B reporting a link of type λ_p with A.

For $\lambda =$ ASYM_LINK, there is no possible proof because B does not hear A.[1]

For $\lambda =$ SYM_LINK, is $\lambda_p =$ ASYM_LINK or SYM_LINK.

For $\lambda =$ SYM_NEIGH or MPR_NEIGH, is $\lambda_p =$ SYM_LINK or SYM_NEIGH.

Regarding TC messages, when A wishes to advertise B as a neighbor, is $\lambda_p =$ SYM_NEIGH or MPR_NEIGH.

4.4 The Certiproof table

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

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

where "originator" is the address of A. The Certificate itself is made up of the remaining fields: as previously said, "address" is the address of B, "link state" is B's link state with respect to A, "timestamp" is the time when A generated the HELLO and ADVSIG messages, "signature" is the signature computed by A on the three fields "address", "link state", and "timestamp".

Please note that, in the implementation, it is obviously not necessary to store the "address" field in the tuple, as it is always the same. However, we present the protocol in this way to be consistent with the Certificate/Proof format and to avoid confusion.

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

[1]This is not a weakness, as all critical operations in OLSR like neighborhood signaling in TCs, MPR selection, etc. concern symmetrical neighbors. Asymmetrical links have the sole purpose to (possibly) establish symmetrical links in an immediate future.

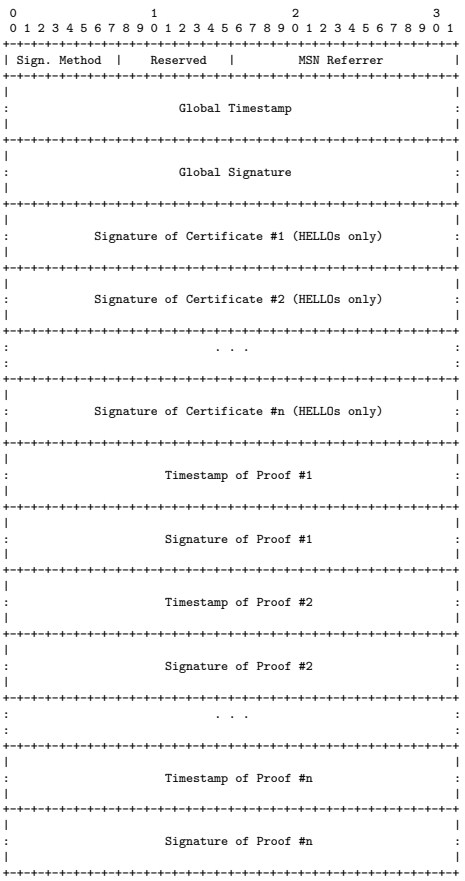

Figure 3: ADVSIG message format.

4.5 The ADVSIG message

The format of this security-enhanced ADVSIG message is shown in figure 3. An ADVSIG message must be generated and sent with every HELLO or TC message. However there is a difference between HELLOs and TCs: while both message types always require Proofs, HELLOs may contain Certificates (i.e., reusable fresh topology information) whereas TCs do not. Hence the `Signature of Certificate` fields exist only in those ADVSIG messages which are coupled to HELLOs.

The `Sign. Method` field specifies which functions are be-

ing used to digest and sign the control message, as well as information about timestamp methods. The `MSN Referrer` field contains the value of the `Message Sequence Number` of the HELLO/TC message with which this ADVSIG message is associated. The `Global Timestamp` is the timestamp of this ADVSIG message and of the HELLO/TC it is coupled with. The `Global Signature` is the signature of the HELLO/TC coupled message, and is computed on the sequence of bits made up of the whole HELLO/TC message[2] and the associated ADVIG message except, of course, the `Global Signature` field itself.

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

The `Timestamp of Proof #`i and `Signature of Proof #`i are the timestamp and signature of the Proof related: to the `Neighbor Interface Address` at position i if the coupled message is a HELLO, or to the `Advertised Neighbor Main Address` at position i if the coupled message is a TC. (The timestamp and signature of the Proof were taken respectively from the `Global Timestamp` and `Global Signature` of a previous HELLO and its accompanying ADVSIG.) If a proof is not required according to section 4.3, the `Timestamp of Proof` and `Signature of Proof` fields are not present.

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

1. the relevant `Neighbor Interface Address` (from the HELLO)
2. the relevant `Link Code` (from the HELLO)
3. the relevant `Global Timestamp` (from the ADVSIG)

Note that in ADVSIG messages sent by A, every `Signature of Certificate` is signed by A, while every `Signature of Proof` is signed by other nodes (which are, or have recently been, neighbors of A).

4.6 The protocol

We denote t_c the current time. δt_1 is the time interval after which a `Global Timestamp` expires. δt_2 is the maximum acceptable time interval between a Certificate and its Proof, after which the Proof is stale and can no longer be used. This means that upon reception of an ADVSIG message, the following conditions must hold for every k: $t_c - \delta t_1 <$ `Global Timestamp` $< t_c$, and `Global Timestamp` $- \delta t_2 <$ `Timestamp of Proof` $k <$ `Global Timestamp`.

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

1. create the HELLO/TC message
2. write t_c in `Global Timestamp`
3. if the message is a HELLO then
   (a) for each `Neighbor Interface Address` i
      i. compute `Signature of Certificate #`i on: `Neighbor Interface Address` i, `Link Code`, and `Global Timestamp`
      ii. if a Proof is required (according to section 4.3) then
         A. find the required Proof (according to section 4.3) in your Certiproof table
         B. copy the Proof's timestamp in `Timestamp of Proof #`i
         C. copy the Proof's signature in `Signature of Proof #`i
4. else if the message is a TC then
   (a) for each `Advertised Neighbor Main Address` j
      i. if a Proof is required then
         A. find the required Proof in your Certiproof table
         B. copy the Proof's timestamp in `Timestamp of Proof #`j
         C. copy the Proof's signature in `Signature of Proof #`j
5. compute the `Global Signature`
6. send the HELLO/TC and ADVSIG messages

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

1. pair off correctly, by matching the `Message Sequence Number` and the `MSN Referrer` fields, the HELLO/TC message with its ADVSIG message
2. check the validity of `Global Timestamp`
3. check the validity of `Global Signature`, using the public key of the node that sent the control message
4. if the message is a HELLO then
   (a) for each `Neighbor Interface Address` k
      i. if a Proof for k was required then
         A. check the validity of `Timestamp of Proof #`k
         B. find $\lambda_1 \leftarrow$ Link Code that should apply to prove Certificate k (according to section 4.3; if more than one Link Code does apply, test all of them)
         C. check the validity of `Signature of Proof #`k computed on: the address of k, λ_1, and `Timestamp of Proof #`k
      ii. if `Neighbor Interface Address` k = your address then
         A. extract (from the HELLO) $\lambda_2 \leftarrow$ `Link Code` relevant to `Neighbor Address` k i.e. your link state
         B. store in your Certiproof table the tuple $\langle$ address of A, your address, λ_2, `Global Timestamp`, `Signature of Certificate #`k $\rangle$
5. else if the message is a TC then
   (a) for each `Advertised Neighbor Main Address` h
      i. if a Proof for h was required then
         A. check the validity of `Timestamp of Proof #`h
         B. find $\lambda_3 \leftarrow$ Link Code that should apply to prove Certificate h

[2]The `TTL` and `Hop Count` fields are considered as set to zero, because they change in transit.

C. check the validity of `Signature of Proof #`h computed on: the address of h, λ_3, and `Timestamp of Proof #`h

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

4.7 Example of the Algorithm

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

1. $A \rightarrow B : \{\varnothing, \varnothing, T_A(t_0)\}_{S_A}$
2. $B \rightarrow A : \{\{\text{“}A : ASYM\_LINK'', T_B(t_1)\}_{S_B}, \varnothing, T_B(t_1)\}_{S_B}$
3. $A \rightarrow B : \{\{\text{“}B : ASYM\_LINK'', T_A(t_2)\}_{S_A}, \varnothing, T_A(t_2)\}_{S_A}$
4. $B \rightarrow C : \{\{\text{“}A : SYM\_LINK'', T_B(t_3)\}_{S_B}, \{\text{“}B : ASYM\_LINK'', T_A(t_2)\}_{S_A}, T_B(t_3)\}_{S_B}$

C now is sure that B's statement about its link state with A is correct. Note the empty Proof for messages in step 1, 2, and 3; this is due to the fact that symmetrical links have not yet been established. To be able to give the Proof in step 4, B stored in its Certiproof table the tuple

$$\langle A, \{\text{“}B : ASYM\_LINK'', T_A(t_2)\}_{S_A} \rangle$$

which was extracted from the data B received from A in step 3.

4.8 Overhead Evaluation

In this subsection we evaluate mathematically the overhead. We assume using 128-bit signatures for the authentication mechanism. We also assume the use of a 32-bit timestamp, which is long enough to contain the time value on a granularity of a 1 ms for a period of more than 49 days. The following evaluations do not include the size of the OLSR packet header.

The size of a HELLO message advertising n nodes varies from $32(n+2)$ to $32(2n+1)$ bits, depending on whether the nodes have the same link state or not. The size of the additional ADVSIG message coupled with this HELLO would have a maximum size (in the scenario in which every entry needs a Proof) of $192 + 288n$ bits.

The size of a TC message advertising n neighbors is $32(n+1)$ bits. The accompanying ADVSIG message would have a size of $192 + 160n$ bits.

A HELLO message requires the computation of $n+1$ signatures by the sender, while a TC requires the computation of just one signature. Both messages require the verification of up to $n+1$ signatures by the receiver.

4.9 Resilience and Remaining Vulnerabilities

With respect to the vulnerabilities explained in section 3, the PKI and the global signature in the ADVSIG message protect against identity spoofing. In this paper, we assume that this security barrier is broken and an attacker has been able to compromise a node or to access its private key. This attacker may perform identity spoofing but cannot try the link spoofing attack. This is an important point. A compromised node can no longer choose the (false) routing information to issue, because this information has to be validated by previous routing information issued beforehand. Hence the network is now robust against one lone attacker. The network is protected even if there are more than one attacker at the same time, provided that they cannot communicate (to issue false Certificates each other) between them.

As mentioned, our solution does not address protection against denial of service attacks, which are difficult to circumvent even in a wired network, or against wormholes.

5. CONCLUSIONS AND FURTHER WORKS

In this paper we have proposed a mechanism to improve the security of the OLSR routing protocol against external attackers. More specifically, we assumed that a mechanism for message signing and sender authentication is deployed, and we handled the case in which an attacker compromises an otherwise trusted node, either by physically tampering with the node's hardware or by stealing the node's private key. Our solution is based on recording recent routing information (HELLO messages) and reusing this information to prove the link state of a node at a later time. This is obtained via a new ADVSIG control message. The overhead of this solution, evaluated mathematically, consists of a maximum of $192 + 288n$ additional bits for each HELLO sent, and $192 + 160n$ additional bits for each TC sent – where n is the number of advertised links or neighbors. Although quite costly in terms of overhead, this mechanism offers the advantage of securing the network against some of the main attacks coming from a compromised node, or from several compromised nodes which are not in direct communication. Further studies about the possible weaknesses of this new system, and simulations to estimate all aspects of overheads, are in our research agenda.

6. REFERENCES

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

[2] Kirk A. Bradley, Steven Cheung, Nick Puketza, Biswanath Mukherjee, and Ronald A. Olsson. Detecting disruptive routers: A distributed network monitoring approach. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 1998.

[3] Sonja Buchegger and Jean-Yves Le Boudec. Performance analysis of the CONFIDANT protocol (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks). In *Proceedings of MOBIHOC*, EPFL Lausanne, Switzerland, June 9–11 2002.

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

[5] T. Clausen (ed) and P. Jacquet (ed). Optimized link state routing protocol (OLSR), October 2003. RFC 3626, Experimental.

[6] Yih-Chun Hu, David B. Johnson, and Adrian Perrig. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002)*, pages 3–13, Calicoon, NY, USA, June 2002.

[7] 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.

[8] 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.

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

[10] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. IEEE INMIC, 2001. Hipercom Project, INRIA Rocquencourt.

[11] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The dynamic source routing protocol for mobile ad hoc networks (DSR), February 24 2003. Internet-Draft, draft-ietf-manet-dsr-08.txt.

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

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

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

[15] S. Murphy, M. Badger, and B. Wellington. OSPF with digital signatures, June 1997. RFC 2154, Experimental.

[16] National Institute of Standards and Technology. Digital Signature Standard (DSS), 27 January 2000. FIPS PUB 186-2.

[17] R. Ogier, F. Templin, and M. Lewis. Topology dissemination based on reverse-path forwarding (TBRPF), February 2004. RFC 3684, Experimental.

[18] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (AODV) routing, July 2003. RFC 3561, Experimental.

[19] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244. ACM Press, 1994.

[20] Ricardo Staciarini Puttini, Ludovic Me, and Rafael Timóteo de Sousa. Certification and authentication services for securing manet routing protocols. In *Proceedings of the Fifth IFIP TC6 International Conference on Mobile and Wireless Communications Networks*, Singapore, October 2003.

[21] Amir Qayyum, Laurent Viennot, and Anis Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. Technical report, Hipercom Project, INRIA Rocquencourt, 2000. INRIA RR-3898.

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

[23] Kimaya Sanzgiri, Bridget Dahill, Brian Neil Levine, Clay Shields, and Elizabeth M. Belding-Royer. A secure routing protocol for ad hoc networks. In *Proceedings of the 10th IEEE International Conference on Network Protocols*, pages 78–89. IEEE Computer Society, 2002.

[24] Manel Guerrero Zapata. Secure ad hoc on-demand distance vector (SAODV) routing, October 2002. Internet-Draft, draft-guerrero-manet-saodv-00.txt.