# Router and Link Admittance Control in the Optimized Link State Routing Protocol version 2 (OLSRv2)

Thomas Clausen, Ulrich Herberg
Hipercom@LIX – Ecole Polytechnique – France
Email: thomas@thomasclausen.org, ulrich@herberg.name

***Abstract*—This paper presents security mechanisms for router and link admittance control in OLSRv2. Digitally signing OLSRv2 control messages allows recipient routers to – individually – choose to admit or exclude the originating router for when populating link-state databases, calculating MPR sets etc. By additionally embedding signatures for each advertised link, recipient routers can also control admittance of each advertised link in the message, rendering an OLSRv2 network resilient to both identity-spoofing and link-spoofing attacks.**

**The flip-side of the coin when using such a link-admittance mechanism is, that the number of signatures to include in each OLSRv2 control message is a function of the number of links advertised. For HELLO messages, this is essentially the number of neighbor routers, for TC messages, this is the number of MPR Selectors of the originator of the message. Also, upon receipt of a control message, these signatures are to be verified. This paper studies the impact of adding a link-admittance control mechanism to OLSRv2, both in terms of additional control-traffic overhead and additional in-router processing resources, using several cryptographic algorithms, such as RSA and Elliptic Curve Cryptography for very short signatures.**

***Index Terms*—OLSRv2, MANET, security, router, link admittance control, digital signatures**

## I. Introduction

Network integrity in routed networks is largely preserved by physically controlling access to the communications channel between routers: know thy peers, trust thy peers — and be able to disconnect thy peers if they are not worthy of the trust, *e.g.* if the topology they present does not match expectations, *i.e.*, routing integrity is protected by admitting only trusted peers, assuming that these, once admitted, are well behaving.

In a MANET (Mobile Ad hoc NETwork), often operated over wireless interfaces, this is less obvious: physical access to the media between routers is not delimited by a cable, but is available to anyone within transmission range; the network topology is time-varying, either due to router mobility or due to time-varying characteristics of the channel – consequently, determining that a peer does not present an "expected topology" and subsequently "disconnecting" it is difficult. As such, MANETs do not introduce particularly new security issues for routing protocols, but rather render existing security issues easier to exploit and, therefore, require re-examining countermeasures for routing protocol resilience.

### A. OLSRv2 Overview

The Optimized Link State Routing Protocol version 2 (OLSRv2) [1], [2], [3], [4], [5] is a successor to the widely deployed OLSR [6] routing protocol for MANETs. OLSRv2 retains the same basic algorithms as its predecessor, however offers various improvements, *e.g.* a modular and flexible architecture allowing extensions, such as for security, to be developed as add-ons to the basic protocol. OLSRv2 contains three basic processes: Neighborhood Discovery, MPR Flooding and Link State Advertisements. The basic operation of OLSRv2 is illustrated in figure 1. Ignoring the gray router *X*, the different elements of OLSRv2, the processes for *Neighborhood Discovery*, *MPR Flooding*, and *Link State Advertisement*, are detailed below. This is followed by a description of the flexible message format used by OLSRv2, as well as the inherent extensibility specifically enabling extensions such as those developed in this paper.

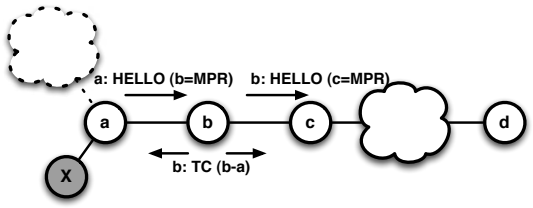


Fig. 1. Basic OLSRv2 operation.

*1) Neighborhood Discovery:* The process, whereby each router discovers the routers which are in direct communication range of itself (1-hop neighbors), and detects with which of these it can establish bi-directional communication. Each router sends HELLOs, listing the identifiers of all the routers from which it has recently received a HELLO, as well as the "status" of the link (HEARD, verified bi-directional – called SYM). A router *a* receiving a HELLO from a neighbor *b* in which *b* indicates to have recently received a HELLO from *a* considers the link *a-b* to be bi-directional. As *b* lists identifiers of all its neighbors in its HELLO, *a* learns the "neighbors of its neighbors" (2-hop neighbors) through this process. HELLOs are sent periodically, however certain events may trigger non-periodic HELLOs.

*2) MPR Flooding:* The process whereby each router is able to, efficiently, conduct network-wide broadcasts. Each router designates, from among its bi-directional neighbors, a subset (MPR set) such that a message transmitted by the router and relayed by the MPR set is received by all its 2-hop neighbors (*i.e.*, the MPR set *"covers"* all 2-hop neighbors). MPR selection is encoded in outgoing HELLOs. The set of routers having selected a given router as MPR is the MPR-selector-set of that router. A study of the MPR flooding

algorithm can be found in [7].

*3) Link State Advertisement:* The process whereby routers are determining which link state information to advertise through the network. Each router must advertise links between itself and its MPR-selector-set, in order to allow all routers to calculate shortest paths. Such link state advertisements, carried in TC messages, are broadcast through the network using the MPR Flooding process. As a router selects MPRs only from among bi-directional neighbors, links advertised in TCs are also bi-directional. TC messages are sent periodically, however certain events may trigger non-periodic TCs. In order to be able to discriminate between fresh and stale information, Link State Advertisements, emitted by a given router, include a sequence number incremented each time that router changes the set of links advertised.

*4) Flexible Message Format:* OLSRv2 employs the format specified in [2], for all protocol messages. This format enables scope-limited message flooding by way of `<hop-limit>` and `<hop-count>` message header fields, modified each time a message is forwarded. The message body format enables compact (aggregated) address representation, also of non-contiguous network addresses, by way of address blocks, and has the ability to associate any number of arbitrary attributes to each such address, by way of inclusion of Type-Length-Value objects (TLVs), referencing the address to which they correspond. Such TLVs are denoted *"Address Block TLVs"*[1]. An example of an attribute that may be associated with an address in OLSRv2, is Link Status = SYM in HELLOs, to indicate that a link between the originator of the message and the indicated address has been verified to be bi-directional. Another example of such an attribute, associated by an OLSRv2 router to specific addresses in HELLO messages is an MPR TLV, indicating a router's MPR selection.

Furthermore, the message body can contain any number of arbitrary attributes not specifically associated to any address, this also by way of inclusion of TLVs. Such TLVs are denoted *"Message TLVs"*. An example of an attribute that may be included in a message in OLSRv2, and which is not associated with any address, is the sequence number included in TCs.

The TLV structure permits any given message to be parsed correctly by allowing an implementation to "skip over" TLVs not recognized, thus enabling extensions to be developed that embed information into existing OLSRv2 control messages.

*5) Inherent Protocol Extensibility:* [4], [5] are conceived to enable protocol extensions to be developed for OLSRv2. This is, in addition to the message format described above, accomplished by allowing that subsequent to the usual control message (HELLO and TC) generation, outgoing messages can be handed off to a protocol extension for further processing. Amongst other things, such an extension can insert addresses, Address Block TLVs and Message TLVs. Moreover, upon receipt of a control message, and prior to the usual processing according to that message type, incoming messages can be processed by a protocol extension – including processing of information from that message (extension specific TLVs, for example), as well as allowing a protocol extension to identify the received message as malformed, and thus prohibit processing of that message by OLSRv2.

[1]In order to avoid repetition of attributes, an Address Block TLV can reference a single, a range or all addresses in a given address block, see [2] for details.

### B. OLSRv2 Vulnerability Taxonomy

As link state protocol, OLSRv2 assumes that (i) each router can acquire and maintain a topology map, accurately reflecting the effective network topology; and (ii) that the network converges, *i.e.* that all routers in the network will have sufficiently identical topology maps. Network connectivity can be disrupted by causing either of these assumptions to not hold, specifically (a) routers may be prevented from acquiring a topology map of the network; (b) routers may acquire a topology map, which does not reflect the effective network topology; and (c) two or more routers may acquire inconsistent topology maps.

In OLSRv2, this translates into that: (i) the links designated by HELLOs to be advertised in TCs reflect actual links in the network; (ii) that the TCs advertise these actual links; and (iii) that TCs are correctly relayed, *i.e.* that the MPR flooding process operates correctly. [8] provides a detailed security analysis of OLSRv2, observing how, and with which consequences, a disruptive attack might be conducted against an OLSRv2 network. A common, and not surprising, observation from [8] is, that *identity spoofing* and *link spoofing*, *i.e.*, that a router in its control traffic either pretends to have the identity of another router or pretends to have (non-existing) links to another router, are major vectors for disruptive attacks on an OLSRv2 network.

### C. Problem Statement

Returning to figure 1, router *a* selects *b* as MPR in order to cover *c*. Router *b*, therefore, advertises the link *b-a* in TCs, throughout the network. If a malicious router, *X* (gray circle) is a neighbor of *a* and spoofs the identity of *c* (more generally, of all neighbors of b), then *a* will not select *b* as MPR. This has as consequences that (i) *b* will not advertise *b-a*; and (ii) the MPR flooding process is disrupted: TCs transiting through *a* will not be relayed by *b* to reach the right-hand side of the network. This is an illustration of the effect of *identity spoofing*.

A possible countermeasure to such an identity spoofing attack is for a protocol extension to admit only control messages originating from routers, whose identity can be verified to not be spoofed, for processing by OLSRv2 – *router admittance control*.

Router admittance control assumes a *transitive trust relationship* between routers: *d* receiving a TC from *b* declaring a link *b-a*, and which *d* (by way of a router admittance control protocol extension) is able to verify was indeed sent from *b*, will have to trust that *b* is correctly behaving (*i.e.*, has not been compromised) and that *b* has properly verified the identity of *a* (the "other end of the link", advertised in the TCs received from *b*) as well as the properties associated herewith. Router admittance control does not permit the recipient of a TC to verify that the content of the TC is valid.

Still in figure 1, should *b* be malicious or compromised, but still in possession of credentials to generate TCs which pass verification by a router admittance protocol extension, it might in its TCs also advertise a fictitious link *b-d*. *c* would receive this and, thus, transit traffic destined for *d* via *b*, rather

than through "to the right" to where *d* is located. This is an illustration of the effect of *link spoofing*.

A possible countermeasure to such a link spoofing attack is for a protocol extension to admit only links, where it can be verified by the recipient that both ends have *"signed off"* for the existence of that link – *link admittance control*.

### D. Paper Outline

The remainder of this paper is organized as follows: Section II describes a basic router admittance control mechanism for OLSRv2. Section III introduces a link admittance control mechanism, allowing per-link verification without assuming transitive trust, also for OLSRv2. Section IV provides a specification of the protocol extension, notably the TLVs and their content, necessary for enabling router and link admittance control, and how the proposed extensions integrate into the OLSRv2 protocol architecture. As the protocol extensions proposed in this paper rely on cryptographic signatures, section V briefly discusses the applicability of shared and public key cryptographic systems for this purpose, and section VI discusses the use of timestamps in the protocol extensions proposed. Section VII studies the performance of the proposed security mechanisms, with particular emphasis on (i) control traffic overhead incurred, and (ii) additional in-router resource requirements, as a consequence of these security mechanisms. This paper is concluded in section VIII.

## II. Router Admittance Control

Router admittance control in OLSRv2 is enabled in [4], [5] by allowing a protocol extension to, upon receipt of a control message and prior to the usual processing hereof, determine if the message originates from a router using a "spoofed identity". Thus, each router must be able to include sufficient credentials in each control message to allow a recipient to make such a determination.

To this end, this paper assumes that (i) each router identity (IP address) is also associated with a cryptographic key, (ii) this key is used for generating and including a cryptographic signature in each outgoing control message, and (iii) that this signature is verified by a receiving router, prior to the control message being processed by OLSRv2. The cryptographic signature is carried in control messages by way of a TLV, specified in section IV.

More precisely, for router admittance control, each router will, for each outgoing control message:

- calculate `sign(ownID, TimeStamp, <msg>);` where `<msg>` is the control message, including all headers, but with the mutable fields `<hop-limit>` and `<hop-count>` (if present) set to zero, and `TimeStamp` is current at the time of signature generation;
- add this signature, as well as `TimeStamp`, by way of a Signature-TLV and Timestamp-TLV (section IV), to the control message.

Each router will, for each incoming control message and prior to it being delivered to OLSRv2 for processing:

- verify the included Signature-TLV;
- consider the message as malformed (and, thus, prohibit its processing by OLSRv2) if either of:
  - no Signature-TLV is present in the received message;
  - the verification fails, *i.e.* the signature does not correspond to the message originator and content;
  - if clocks are synchronized and Replay Attacks are of concern, the included `TimeStamp` is "too old" (refer also to section VI).
- otherwise, consider the message as correctly formed according to the Router Admittance Control protocol extension.

If a message is so considered "correctly formed", it implies that the originator of the message either is not "spoofing" its identity – or, that the originator has managed to acquire the credentials, necessary for generating a signature corresponding to a spoofed identity.

## III. Fine-Grained Security: Link Admittance Control

In order to allow a router receiving a control message to verify "both sides" of the link, (i) both sides must be able to establish that the link exists, and (ii) information "signing off" for this must be included in the control message. The TLV format in [2] enables that that information can be associated with each address, advertised in a control message.

For each address (the *other end of the link*) advertised in a control message, the originating router includes a signature embedding its own address, the address of the peer, the timestamp of emission, and any additional attributes that the originating router has associated with that link, by way of TLV inclusion, *e.g.*, if the link is HEARD or SYM (for HELLO messages) or if an address is routable (for TC messages). The signature so included is, thus: `sign(t, ownID, peerID, own-attribute-list)`. The router also signs the message as described in section II, thus notably including its own timestamp in the message as well. Additionally, the router includes the most recent signature previously received for this link from the peer, the corresponding timestamp received from the peer, as well as the attribute list for this link received from the peer (*i.e.*, the information which the peer used for calculating the signature). A router receiving such a message can, then, verify if (i) the two routers agree on the attributes associated with the link, and (ii) do so at approximately the same time[2].

Consider the example depicted in figure 2.

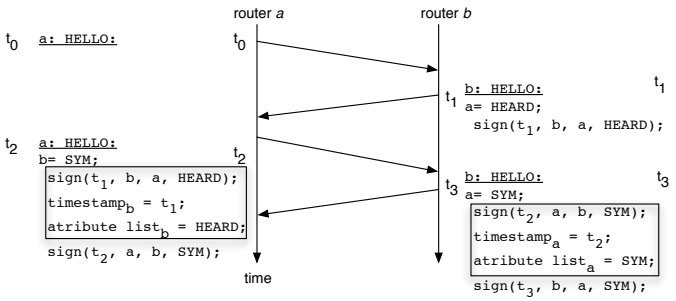


Fig. 2. Example of link admittance control in the Neighborhood Discovery process of OLSRv2: attributes listed in "boxes" are those received from the "peer" in a previous HELLO message.

[2]Timestamps are included to counter replay attacks. Using timestamps requires roughly synchronized system clocks. A similar mechanism using nonces could be possible, when clocks are not assumed to be synchronized.

At $t_0$, router *a* sends a HELLO; it has no neighbors and thus the HELLO is empty. When *b* receives this HELLO, it will – as usual in OLSRv2 – advertise *a* as HEARD in its next HELLO, at $t_1$ and associate its signature `sign(t1, b, a, HEARD)` to this advertisement, by way of a TLV. Any router receiving this HELLO can verify only that *b* claims information about the link *a-b*.

At $t_2$, router *a* will proceed in a similar fashion, advertising *b* as SYM, associating its own signature `sign(t2, a, b, SYM)`. The router will also include the information shown inside the "box": the last received signature for this link, received from *b*, `sign(t1, b, a, HEARD)`, and the information necessary for a third-party to be able to verify the signature of *b*: the timestamp $t_1$, and the attribute list corresponding to this link as received from *b* (HEARD). Any router receiving this HELLO can by verifying the signatures observe that *a* and *b* at this point have claimed different information about *a* (*a* describes the link as SYM, *b* describes it as HEARD).

At $t_3$, router *b* may consider advertising *a* as SYM, and associating its own signature `sign(t3, b, a, SYM)`. The router will also include the information shown inside the "box": the last received signature for this link, received from *a*, `sign(t2, a, b, SYM)` and the information necessary for a third-party to be able to verify the signature of *a*: the timestamp $t_2$, and the attribute list corresponding to this link as received from *a* (SYM). Any router receiving this HELLO can by verifying the signatures observe that at $t_2$ and $t_3$, respectively, both *a* and *b* have claimed claimed that the link *a-b* is symmetric. If $t_2$ and $t_3$ are sufficiently close, a recipient may conclude that a symmetric link exists between *a-b*, and that both *a* and *b* have "signed off" herefore. The link can therefore be considered as "trusted", and thus reflected in the link- and neighbor-sets of OLSRv2.

The main impact, in terms of protocol operation, is that if link admittance control is used when admitting routers to the 2-hop set, then one further HELLO message exchange is required in order for a router to be able to detect 2-hop links as "signed off" as symmetric by both ends.

## IV. Protocol Extension Specification: Router and Link Admittance Control

In the following, the router and link admittance control protocol extension, proposed by this paper, is specified, in particular the TLV types introduced, as well as the interaction between this protocol extension and OLSRv2.

### A. TLV Specification

Three TLVs are required: a timestamp TLV, a signature TLV and an attributes TLV. The timestamp TLV and the signature TLV, both, can be used as Message TLVs (*i.e.* included in the header of a control message) and as Address Block TLVs (*i.e.* associated with one or more addresses in the message body). This section specifies the content of `<value>` in the three proposed TLVs, for use in the format described in [2].

#### 1) Timestamp TLV:

```
<timestamp> := <time-value>
```

where: `<time-value>` contains the timestamp. A timestamp is essentially "freshness information", and may *e.g.* correspond to a UNIX-timestamp, GPS timestamp or a simple sequence number. For the performance study of section VII, the timestamp is a four-byte long integer, counting the seconds from the start of the simulation.

#### 2) Signature TLV:

```
<sign-tlv> := <hash-fkt><sign_algo><sign>
```

where: `<hash-fkt>` and `<sign_algo>` are 1-octet long fields identifying the choice of hash function and signature algorithm, respectively, and `<sign>` contains the digital signature.

#### 3) Attributes TLV:

```
<attributes> := {<attribute-value>}*
```

Recalling that for each address included in a message, two signatures are ultimately included. If the message is originated by router *a* and contains an address of a peer *b*, then for the link *a-b* a signature generated by both *a* and *b* is included. In order for a third-party *c* to be able to verify also the signature from the peer *b*, the exact information over which the peer *b* calculated this signature needs to be available to *c*. In figure 2 at $t_2$, for example, this information is included in the box: the timestamp ($t_1$) and the attribute list (HEARD). This information has been received by *a* by way of TLVs, and the signature (`sign(t1, b, a, HEARD)`) can be verified by *a* using the identities of both routers *a* and *b* as well as the timestamp ($t_1$) from the timestamp TLV and the attribute (HEARD) from the attributes TLV.

### B. OLSRv2 Interaction

Due to the flexible nature of the specification of OLSRv2, the router and link admittance control extension, presented in this paper, can be designed as an independent module. This module is invoked when an incoming control message has been successfully parsed, and before further processing by OLSRv2, as well as whenever an outgoing message is about to be sent. This simple architecture allows combination of other OLSRv2 extensions with this router and link admittance control extension, without encumbering such other extensions. The flow of incoming and outgoing messages between the different modules is depicted in figure 3.

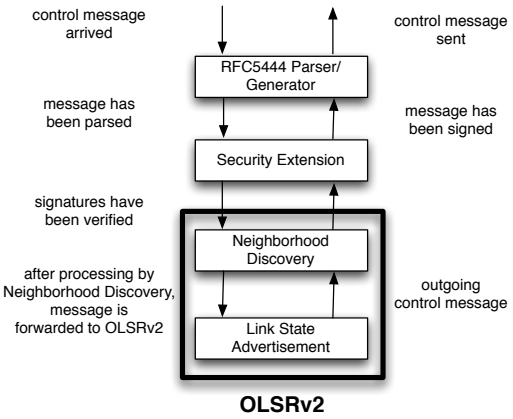


Fig. 3. Message flow between NHDP, OLSRv2 and the security extension.

## V. Cryptographic Keys

Using cryptographic signatures in control messages allows the recipient of a message to (i) verify the integrity upon receipt, (ii) verify if the originator is to be admitted to the network, and (iii) verify the identity of originator of the control message. For (i) and (ii), a "pre-shared secret", such as a secret passphrase or a symmetric key, suffices: only routers possessing the secret are able to correctly sign control messages and addresses within, which allows exclusion of "all but pre-approved routers". However, (iii) requires asymmetric keys for allowing per-principal authentication. As the link admittance control extension, presented in this paper, relies on bi-directional verification of links between routers, per-principal authentication is a requirement.

For assuring the reliability of the admittance control system, it is paramount that the cryptographic keys are only accessible by the router that they are allocated to. Furthermore, it has to be ensured that the keys cannot be derived from any information exchanged between the routers, *i.e.*, that the cryptographic algorithm is not vulnerable to attacks, other than brute-force with sufficient efforts for such a brute-force attack being appropriate for deployment requirements.

## VI. Timestamps

The router and link admittance control extension can provide protection against identity-spoofing and link-spoofing of unadmitted routers (*i.e.* routers not able to correctly sign messages), but malicious routers can still record and replay messages (see [8] for details on such "replay attacks"). These replay attacks can be partly avoided by introducing "freshness" information, such as timestamps or nonces, in messages. A message which is replayed some time after the recording, can be detected as being "old" (at least, when the clocks of the routers are roughly synchronized).

In the protocol extension proposed in this paper, both whole messages and individual links are candidates for "being replayed". Consider a router *X* which has been compromised (*i.e.*, the attacker has access to appropriate credentials to generate correctly signed messages). If no timestamps were included in the per-link signatures, *X* could record such per-link signatures and include them later in its messages, spoofing links to non-existing neighbors.

## VII. Performance Study

This section presents a performance study, by way of simulations with NS2 [9], of the link admittance control mechanism, in comparison to a simple router admittance control mechanism, both in terms of message overhead and CPU time. Simulations have been conducted with JOLSRv2 [10] using relatively standard scenario parameters (1km$^2$ square, 0-300m segments of random walk at 2-8$\frac{m}{s}$, and 0-5s pause-time). The number of routers was varied between 10 and 50, and each value has been averaged over 20 simulation runs. The performance parameters studied are the extra control traffic overhead and the in-router message generation/processing overhead incurred by the mechanisms presented in this paper. JOLSRv2 uses AgentJ [11] for interfacing with NS2. AgentJ/NS2 permits single thread execution, without preemption, allowing instrumenting the signature generation and verification code to record the time spent on each such operation[3]. For the simulations, an Intel Core 2 CPU with 2.1 GHz and 4 GB of RAM was used.

### A. Overhead of Link Admittance Control

Using router admittance control, described in section II, only a single signature is included per control message. Using link admittance control, described in section III, up to two signatures are included per advertised address. Thus, the message size will grow with the density of the network, as depicted in figure 5 using RSA-1024 [12], DSA-1024 [13], ECDSA-160 [14], and HMAC-80 [15][4], with the numbers being the key length in bits for each respective algorithm. For RSA, DSA and HMAC, the implementations directly provided by Java 6 have been used, whereas ECDSA is a custom implementation.

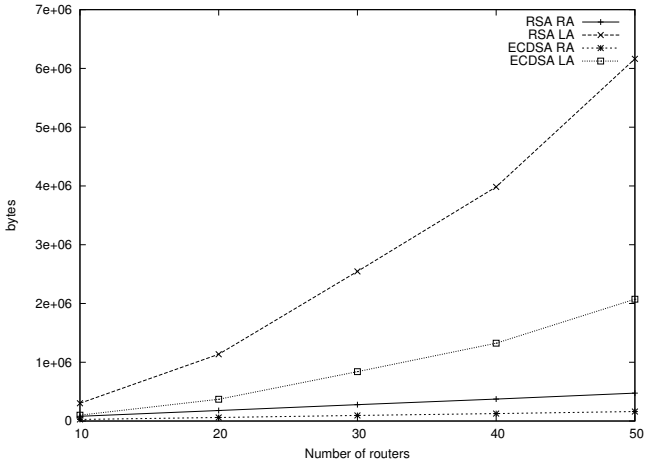


Fig. 4. Total overhead incurring due to signature inclusion.

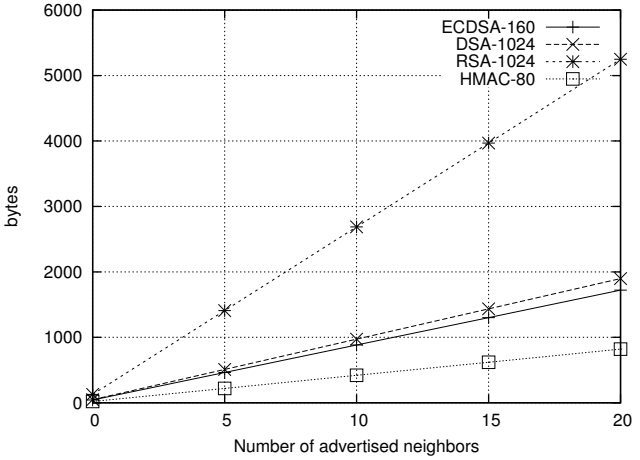


Fig. 5. Link admittance control: Signature overhead per control message with increasing number of advertised neighbors.

[3]For completeness: AgentJ rewrites `System.currentTimeMillis()` such as to return the "simulator time", whereas `System.nanoTime()` is not rewritten and therefore returns the "wall clock time".

[4]Note that HMAC, strictly speaking, is not a signature algorithm, but a Message Authentication Code, see section V.

In the following, only ECDSA and RSA are considered for the comparison, exploring their differences in terms of (i) message overhead and (ii) CPU time for processing and generating signatures.

Figure 4 depicts the cumulative overhead in the network, due to inclusion of message signatures and address signatures. In this figure, as well as in the following, the overhead only considers the size of the signatures, and *not* the content of the HELLO or TC message themselves. Thus, for an unsigned message, the overhead would be 0. For the router admittance control mechanism (denoted "RA"), the per-message overhead is constant, and the cumulative overhead is a function of the number of control messages – itself a function of simulation time and number of routers. Note that the length of the control message does not influence the length of the signature, since the signature is always calculated over an SHA1 hash of the message. With link admittance control (denoted "LA"), the total overhead grows polynomial with increased number of routers and increased density in the network, as up to two signatures are added per advertised neighbor in a control message. As RSA-1024 signatures are longer than the corresponding ECDSA-160 signatures, the total overhead with RSA grows considerably faster.

Using smaller signatures (*e.g.*, as provided by ECDSA) is, in terms of message size, particularly beneficial for link admittance control. Longer signatures (such as RSA) leads to (i) higher bandwidth consumption for control traffic, and therefore (ii) higher energy consumption[5], as well as (iii) the possibility that the IP packet gets fragmented when its size is greater than the MTU of the underlying link layer. This can be well observed in figure 5; assuming an MTU of 1500 bytes (*e.g.* for Ethernet), messages signed with RSA would be fragmented (with the associated risk of any one fragment being lost causing the whole message to be lost) with about five neighbors, whereas ECDSA would allow roughly three times more neighbors.

### *B. In-Router Resource Requirements*

This section analyses the CPU time for creating and parsing signatures in control messages in OLSRv2 using the router admittance and the link admittance control mechanisms respectively.

Figure 6 depicts the cumulative time each router spends, over the duration of 100 seconds, on generating signatures in JOLSRv2, with router admittance (denoted "RA") and link admittance (denoted "LA"), both. For router admittance control, the time each router spends in total on generating signatures is constant, since every router periodically creates HELLO and TC messages, independently of the size of the network. As expected, using link admittance control significantly increases the amount of CPU time required for generating control messages, since the number of signatures per message to be generated increases with the density of the network. ECDSA and RSA have similar time consumption for generating signatures.

[5][16] states *"The energy cost of transmitting 1Kb a distance of 100 meters is approximately 3 joules. By contrast, a general-purpose processor with 100MIPS/W power could efficiency execute 3 million instructions for the same amount of energy"*, indicating that shorter (but more computationally intensive) signatures for certain applications, such as energy constrained devices, may be preferential.

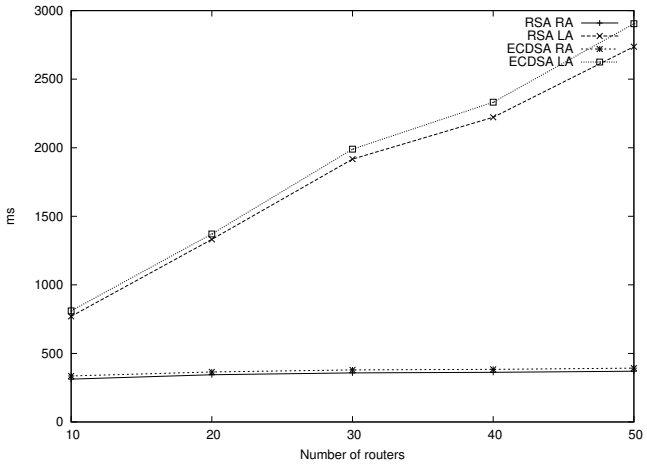


Fig. 6. Cumulative time per router spent on generating message signatures.

The corresponding cumulative processing time in each router is depicted in figure 7. Each router generates HELLOs, which must be processed, and so its signatures verified, by its neighbors. Thus, increasing the network density increases the number of HELLOs that a given router receives and, therefore, the number of signatures to verify. Depending on the network topology and MPR selection, additional routers may also incur additional TCs, whose processing and signature verification is to be conducted by each other router in the network. Link admittance control significantly increases the amount of CPU time spent for verifying signatures for control messages. RSA signatures are very fast to verify, while verifying ECDSA signatures consumes a considerable amount of CPU time. Since every signature has to be verified before it can be forwarded, the total amount of time spent in each router for verifying signatures is considerably higher than for generating messages.

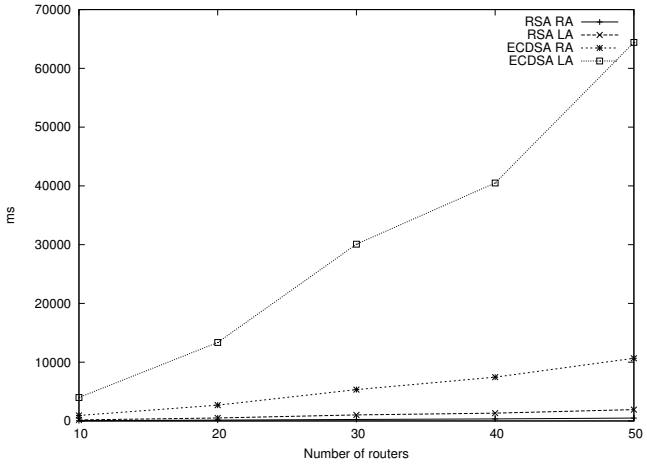


Fig. 7. Cumulative time per router spent on verifying message signatures.

## VIII. Conclusion

When OLSRv2 routers use digitally signed control messages for admittance control, these routers can verify the identity of control message originators and the integrity of the messages. However, a router has to trust the message originator that the advertised links in the HELLO or TC message are valid. This paper specifies a router and link admittance control protocol extension to OLSRv2 , which allows a router to verify each advertised link from incoming control messages, by signing "both ends of the link". The router and link admittance control protocol extension is generic, in that it is not tied to any specific cryptographic system. Indeed, the mechanism operates as long as the choice of cryptographic system allows for per-principal authentication and signature generation.

A performance study of this extension is presented, quantifying the impact in terms of increased control traffic overhead and increased per-message generation and processing time, exemplified by using two relatively common cryptographic systems: RSA, for its performance in verification of signatures, and ECDSA for its short signature lengths for the same "strength" of signatures. It is argued that using shorter signatures may be advantageous when using such a router and link admittance security mechanism, since the additional overhead grows linear with the density of the network. Using longer signatures leads to (i) higher bandwidth consumption for control traffic, and therefore (ii) higher energy consumption, as well as (iii) the possibility that the IP packet gets fragmented when its size is greater than the MTU of the underlying link layer.

It is observed, however, that regardless of the choice of cryptographic system, this router and link admittance control protocol extension is no "free lunch": other than the size increase in control messages, the time required for signature generation and verification is – unsurprisingly – not negligible.


## Acknowledgements

The authors would like to thank Jérôme Milan (LIX, Ecole Polytechnique) for providing the ECDSA implementation and for his assistance with its integration into JOLSRv2.


## References


[1] T. Clausen, C. Dearlove, B. Adamson, "RFC5148: Jitter Considerations in Mobile Ad Hoc Networks (MANETs)", Informational, http://www.ietf.org/rfc/rfc5148.txt

[2] T. Clausen, C. Dearlove, J. Dean, C. Adjih, "RFC5444: Generalized Mobile Ad Hoc Network (MANET) Packet/Message Format", Std. Track, http://www.ietf.org/rfc/rfc5444.txt

[3] T. Clausen, C. Dearlove, "RFC5497: Representing Multi-Value Time in Mobile Ad Hoc Networks (MANETs)", Std. Track, http://www.ietf.org/rfc/rfc5497.txt

[4] T. Clausen, C. Dearlove, J. Dean, "I-D: MANET Neighborhood Discovery Protocol (NHDP)", Work In Progress, http://tools.ietf.org/id/draft-ietf-manet-nhdp-12

[5] T. Clausen, C. Dearlove, P. Jaquet, "I-D: The Optimized Link State Routing Protocol version 2 (OLSRv2)", Work In Progress, http://tools.ietf.org/id/draft-ietf-manet-olsrv2-11

[6] T. Clausen, P. Jacquet, "RFC3626: Optimized Link State Routing Protocol (OLSR)", Experimental, http://www.ietf.org/rfc/rfc3626.txt

[7] A. Qayyum, L. Viennot, A. Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks", 35th Annual Hawaii International Conference on System Sciences (HICSS'2001)

[8] U. Herberg, T. Clausen, "Security Issues in the Optimized Link State Routing Protocol version 2 (OLSRv2)", International Journal of Network Security & Its Applications (IJNSA), Volume 2, Number 2, 2010

[9] http://www.isi.edu/nsnam/ns

[10] U. Herberg, "JOLSRv2 – An OLSRv2 implementation in Java", Proceedings of the 4th OLSR Interop workshop, October 2008

[11] U. Herberg, I. Taylor, "Development Framework for Supporting Java NS2 Routing Protocols", Proceedings of the 2010 International Workshop on Future Engineering, Applications and Services (FEAS), May, 2010

[12] B. Kaliski, J. Staddon, "PKCS 1: RSA Cryptography Specifications Version 2.0", Informational, http://www.ietf.org/rfc/rfc2437.txt

[13] National Institute of Standards & Technology, "Digital Signature Standard", NIST, FIPS PUB 186-3, June 2009

[14] D. Johnson, A. Menezes, S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)", International Journal of Information Security, Volume 1, Number 1 / August, pages 36-63, 2001

[15] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", Informational, http://www.ietf.org/rfc/rfc 2104.txt

[16] G. J. Pottie, W. J. Kaiser, "Wireless integrated network sensors", Communications of the ACM, volume 43, number 5, page 51-58, 2000.