# Digital Signatures for Admittance Control in the Optimized Link State Routing Protocol version 2

Ulrich Herberg, Thomas Clausen, Jerome Milan
Laboratoire d'Informatique (LIX)
Ecole Polytechnique, France
Email: Ulrich@Herberg.name, Thomas@ThomasClausen.org, Jerome.Milan@lix.polytechnique.fr

*Abstract*—**Public community Mobile Ad Hoc NETworks (MANETs), such as the "Funkfeuer" or "Freifunk" networks, scale up to several hundreds of routers, connecting users with each other, and with the Internet. As MANETs are typically operated over wireless channels (*e.g.* WiFi), access to these networks is granted to anyone in the radio range of another router in the MANET, and running the same MANET routing protocol. In order to protect the stability of the networks from malicious intruders, it is important to ensure that only trusted peers are admitted to participate in the control message exchange, and to provide means for logically "disconnecting" a non-trustworthy peer.**

**This paper presents the concept of admittance control for the Optimized Link State Routing Protocol version 2 (OLSRv2), and suggests a security extension based on digital signatures. Due to the flexible message format of OLSRv2, this extension keeps compatibility with the core OLSRv2 specification. Several standard digital signature algorithms (RSA, DSA, ECDSA), as well as HMAC, are compared in terms of message overhead and CPU time for generating and processing 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. Routing integrity is thus 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 counter-measures for routing protocol resilience.

### *A. OLSRv2 Overview*

The Optimized Link State Routing Protocol version 2 [1], [2], [3], [4], [5] is the successor of the widely used OLSR [6] routing protocol for MANETs. A proactive link state protocol, OLSRv2 retains the same basic algorithms as its predecessor,

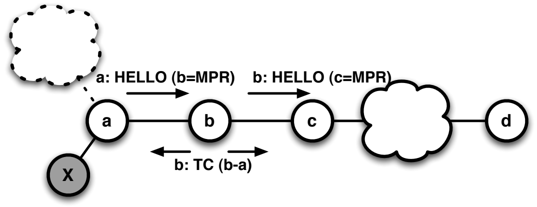

Figure 1. Basic OLSRv2 Operation

but offers various improvements, *e.g.* a modular and flexible architecture allowing extensions to be developed as add-ons to the basic protocol.

OLSRv2 contains three basic processes: Neighborhood Discovery, Link State Advertisement and MPR Flooding. Neighbor Discovery and Link State Advertisement manifest themselves through HELLO and TC messages, generated periodically. HELLOs and TCs, both, advertise links between the router generating the message and (a subset of) its neighbors – possibly also indicating additional information pertaining to each such link. HELLOs are used for detecting bi-directionality of links, and list all neighbors, each associated with a status flag (HEARD, SYM). HELLOs are exchanged only with direct neighbors. TCs are used for sharing link-state information network-wide, and list only neighbors with which a symmetric link has been identified (by the Neighborhood Discovery process). TCs include a sequence number, used to allow recipients to exclude outdated information, incremented when the set of advertised links changes. TCs are delivered through the network using MPR Flooding.

MPR flooding is a process for each router to efficiently perform network-wide broadcasts. Each router designates, a subset (MPR set) from among the neighbors with which a bi-directional link has been identified, such that a message transmitted by the router and relayed by the MPR set is received by all its 2-hop neighbors. MPR selection is signaled by a router by associating a flag (MPR) to the selected neighbors 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].

MPR selection further serves to allow designation of which links are to be included in TC messages: a router must (at least) include links to all its MPR-selector-set. Thus, the schematic operation of OLSRv2 is illustrated in figure 1.

### B. *OLSRv2 Security Problem Statement*

Correct operation of OLSRv2 assumes that each router is able to acquire a consistent topology map, reflecting the effective network topology. In OLSRv2, this implies 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.

In figure 1, router *a* selects *b* as MPR in order to cover *c*. *b*, therefore, advertises the link *b-a* in TCs. If *a*, in its HELLOs, "pretends to be" (spoofs the identity of) *d*, then *b* will instead advertise the link *b-d*. If *b* spoofs the identity of *d* when generating TCs, the link *d-a* is advertised. As neither *b-d* nor *d-a* exist in the network, this will cause incorrect topology maps in routers – with potential consequences including routing loops, parts of the network being disconnected or traffic to the same destination from different parts of the network terminating in different routers [8]. 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.

Symmetric attacks exist by instead of spoofing the *identity of itself*, a router spoofs the *identity of its neighbors – i.e* spoofs links in the network: *e.g.*, if *b* sends TCs advertising a link *b-d* without having a neighbor spoofing the identity of *d*, this will have similar effects.

### C. *Paper Outline*

The remainder of this paper is organized as follows: section II presents the general idea of "admittance control" in OLSRv2, allowing a router to employ cryptographic signatures to detect, and take proper actions, against identity spoofing. Section III expresses this in terms of an extension to OLSRv2, and section IV studies the performance of an OLSRv2 network when using this extension and with various cryptographic algorithms. [9] 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. This paper , therefore, in place of the usual network performance metrics (overhead, bandwidth, ...) studies per-message processing overhead, both for when generating and receiving messages. The paper is concluded in section V.

## II. Admittance Control

The objective in this paper is to provide a mechanism, akin to that which is employed in "classic" routed networks as described in I: a way of ensuring that only trusted peers are admitted to participate in the control message exchange between routers – and, thus, also a way for a router to logically "disconnect" a non-trustworthy peer. Absent the ability to physically control access to the channel between routers, the mechanism employs logical "admittance control" by including cryptographic signatures in control messages (HELLO and TCs), and requires their verification prior to accepting a control message for processing or forwarding.

Verification of a cryptographic signature, associated with a control message, allows the recipient to assert that (i) the control message has not been altered while in transit and (ii) the originator of the control message is in possession of a cryptographic key for generating a valid signature. Concerning (ii), a cryptographic algorithm employing symmetric shared keys supports simple discrimination between "trusted" and "untrusted" routers. Employing asymmetric keys enables discrimination between individual trusted routers, thus prohibiting a "trusted router" from spoofing the identity of another router (trusted or not) in the network.

Associating signatures with control messages is not without effect on network performance: generation and verification of signatures incurs a computational overhead in each router, and inclusion of signatures in control messages increase the number of bits to transmit over the wireless interface. These effects are, for a selection of cryptographic algorithms, quantified in this paper .

### A. *Resilience Evaluation*

The admittance control mechanism enables routers to determine if a received control message originates from a "trusted router", discarding information received from non-trusted routers. This leaves two remaining vulnerabilities uncovered: (i) recording control traffic from a "trusted router" for later replaying (possibly elsewhere in the network) and (ii) trusted routers misbehaving, *e.g.* by spoofing links.

For (i), a countermeasure in the form of synchronized time-stamps can be employed, included in each control message; for (ii), inclusion of signatures for both ends of an advertised link can be considered. Synchronized time-stamps and per-link signatures are, however, the subject for a subsequent article.

## III. Specification

Introducing admittance control to OLSRv2 requires specification of (i) a way of associating signatures to control messages and (ii) the necessary processing for generating and verifying signatures (and, appropriate actions to take in case verification fails). Compared to its predecessor, OLSRv2 facilitates this through the use of [2] as packet and message format, and through explicit "hooks" in [4], [5] for recognizing external reasons for rejecting a message as malformed.

### A. *Signature TLV Structure*

[2] enables a generic way for protocol extensions to add information to control messages by way of inclusion Type-Length-Value (TLV) objects. Thus, a security extension can associate a signature to a HELLO or TC by including a "Signature TLV" in the control message:

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

where: `<hash-fkt>` and `<sign_algo>` identify the choice of hash function and signature algorithm, respectively, and `<sign>` contains the digital signature, calculated thus:

```
sign = sign_algo(hash-fkt(message))
```

Verification of a message is a boolean operation, acting as a black-box for the routing protocol and returning true if the message signature verifies, false otherwise:

```
verified = verif(message, <sign-tlv>)
```

### B. Message Generation and Processing

As enabled by [4], [5], subsequent to the usual HELLO and TC generation, outgoing messages are signed and a `<sign-tlv>` included. Upon receipt of a HELLO or TC, a message is identified as malformed and, thus, not processed, if `verif(...)` for that message returns false. For TC messages, per-hop mutable fields (hop-count and hop-limit) are set to 0 for calculating `<sign>` and `verif(...)`.

Note that due to the packet and message format of OLSRv2 [2], using this signature extension keeps compatibility with the "core" OLSRv2 specification. A router, that uses OLSRv2 without the extension, does not know the TLV type and will consequently ignore the information contained in these TLVs. It it thus able to correctly process signed control messages, however, its HELLOs and TCs will be rejected by all routers applying the security extension, since the messages are not signed.

## IV. Performance Study

Introduction of signatures provides a measure of "admittance control" to OLSRv2 – at the expense of increased control message sizes and per-message-generation/processing overhead, dependent on the cryptographic algorithms employed. This additional overhead is quantified by way of NS2 simulations, using with a selection of cryptographic algorithms: RSA-1024, DSA-1024 and ECDSA-160. The signature-lengths in these algorithms have been chosen so as to provide similar resilience against attacks, with an 80-bit security level [10].

RSA [11], DSA [12] and ECDSA [13] employ asymmetric keys, *i.e.* permit by verification of the signature to also verify the identity of the router generating the message. For comparison, the symmetric HMAC-80[1][14] algorithm is also included. The results presented in this section are all normalized wrt. OLSRv2, *i.e.* only the additional overhead incurring from the security extension is depicted.

Simulations have been conducted using JOLSRv2 [15] using relatively standard scenario parameters (1km$^2$ square, 100m segments of random walk at 2-8$\frac{m}{s}$, 0-5s pause-time and 5 concurrent CBR streams of 3.2kb/s). The number of routers was varied between 10 and 50, and each value has been averaged over 15 simulation runs.

[1]Note that HMAC, strictly speaking, is not a signature algorithm, but a Message Authentication Code. As such, it does not offer per-principal authentication nor non-repudiation. In the rest of the paper , we will not make this distinction and will abusively refer to HMAC as a signature method.

| Crypto | Generation time | Processing time | Overhead |
|---|---|---|---|
| ECDSA-160 | 2.11 ms | 3.73 ms | 42 bytes |
| DSA-1024 | 2.78 ms | 5.26 ms | 46.31 bytes |
| RSA-1024 | 4.84 ms | 0.33 ms | 128 bytes |
| HMAC-80 | 0.07 ms | 0.04 ms | 20 bytes |

Table I
SIMULATION RESULTS: PER-MESSAGE GENERATION/ PROCESSING TIME AND OVERHEAD

In addition to the additional control traffic overhead, the in-router message generation/processing overhead has also been measured. JOLSRv2 uses AgentJ [16] for interfacing with NS2. AgentJ/NS2 permits a single thread execution at a time, without preemption, thereby allowing instrumenting the signature generation and verification code to record the time spent on each such operation[2]. For the simulations, an Intel Core 2 CPU with 2.1 GHz and 4 GB of RAM was used.

### A. Simulation Results

Table I summarizes the average measured time for generating and processing control messages, as well as the average overhead, incurred by each of the four signature processes. For OLSRv2 without signatures, these values would all be zero.

As a first observation, HMAC-80 requires significantly less time than ECDSA, DSA and RSA for generating a message signature. For verification of a message signature, HMAC likewise spends substantially less time than ECDSA and DSA, whereas RSA is close to HMAC in the time it requires for a verification. Verification of RSA signatures is faster than both ECDSA and DSA, however with much greater overhead as well. These figures are hardly surprising; the characteristics of the various signature algorithms are well-known [12].

Figure 2 depicts the cumulative time each router spends, over the duration of 100 seconds, on generating message signatures in JOLSRv2. As the number of routers grow in the network, so does the time each router spends in total on generating signatures. The reason for this is, that OLSRv2 enables "triggered message generation", *i.e.* that message generation may result from some external event, in addition to periodical messages. Thus, with more routers being present in the network, more links are detected as appearing/disappearing – and therefore more messages, and corresponding signatures, are generated.

The corresponding cumulative processing time in each router is depicted in figure 3. Each router generates HELLOs, which must be processed 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. Finally, as indicated above, triggered messages as more links appear/disappear in the network also cause more messages with signatures to verify.

[2]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".

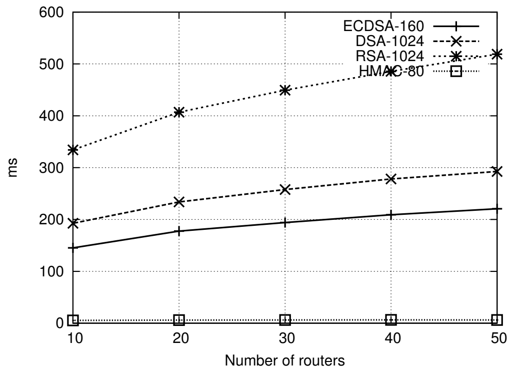

Figure 2. Cumulative time per router spent on generating message signatures.

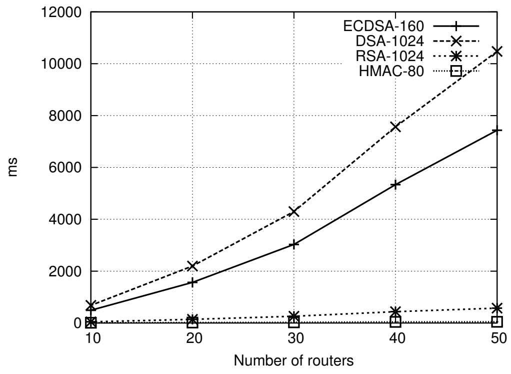

Figure 3. Cumulative time per router spent on verifying message signatures.

A couple of general observations can be made concerning the results. While HMAC and RSA do not need a significant amount of time for signature verification, ECDSA routers spend almost 8% of their time verifying signatures, while DSA spends more than 10%. This might still appear reasonable – but it has to be kept in mind that the simulation was performed on relatively fast hardware[3], whereas devices in an OLSRv2 network may be far less powerful than that, such as smartphones or even sensors. Thus, one might be tempted to use RSA signatures if CPU ressources is the primary concern and ECDSA otherwise. That said, ECDSA scales better than RSA (*e.g.*, increasing from 80-bit security level to 128-bit level will lengthen ECDSA keys by about 40% and RSA keys by 200%), which should be taken into account when elevating the security level.

Figure 4 depicts the cumulative overhead in the network, due to the inclusion of message signatures. Intuitively, as the per-message overhead is constant, the cumulative overhead would be a function of the number of control messages – itself a function of simulation time and number of routers. That the curves in figure 4 grow slightly faster than linearly with the number of routers is, as stated previously, due to triggered messages.

## V. Conclusion

This paper has presented a general extension to OLSRv2 for providing "admittance control" in an OLSRv2 network: enabling admitting "trusted routers" and excluding "non-trusted" routers from participating in the control message exchange between routers, thereby providing a mode-of-operation of an OLSRv2 network, similar to that which is otherwise employed for preserving network integrity in routed networks.

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. The signature algorithms used were RSA-1024, DSA-1024 and ECDSA-160 – chosen and parameterized in order to provide for comparable resilience.

[3]Albeit using the less than optimal Java multi-precision arithmetic.

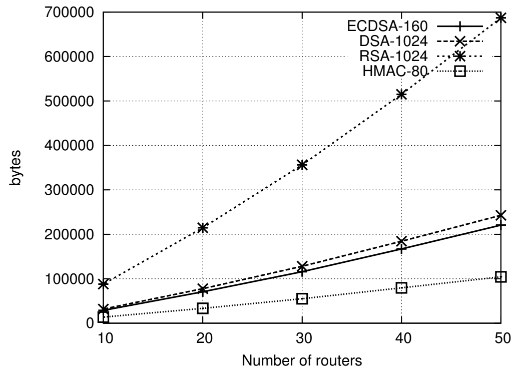

Figure 4. Total overhead incurring due to signature inclusion.

## 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

[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

[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] T. Clausen, U. Herberg, "Vulnerability Analysis of the Optimized Link State Routing Protocol version 2 (OLSRv2)", INRIA Research Report 7203, February 2010

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

[10] National Institute of Standards & Technology, "Recommendation for Key Management – Part 1: General", NIST, Special Publication 800-57, March 2007

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

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

[13] 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

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

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

[16] U. Herberg, "Integrating Java Support for Routing Protocols in Ns2", INRIA research report 7075, October 2009