UNIVERSITÀ DEGLI STUDI ROMA TRE

Facoltà di Ingegneria
Corso di Laurea in Ingegneria Informatica

Tesi Di Laurea

Trusted Routing in OLSR MANETs

Laureando
Claudio Pisa
Matricola 60710

Relatore
Giuseppe Di Battista
Università degli Studi
Roma Tre

Correlatore
Giuseppe Bianchi
Università degli Studi di Roma
Tor Vergata

Anno Accademico 2007-2008

Ringraziamenti

A tutta la mia famiglia, in particolare mia madre e mio padre, per avermi supportato in tutte le mie scelte, a Simona, per tutto quello che fa per me, a tutta ninux.org, per avermi fatto crescere, e specialmente a Nino, per avermi trascinato nel progetto e spinto ad imparare cose nuove, a tutto il laboratorio del Netgroup di Tor Vergata, in particolare ai miei co-correlatori Saverio e Simone, a tutti quanti i miei compagni di università, per i bei momenti passati insieme, ai ragazzi di C-base e di Freifunk, per quello che fanno, per l'atmosfera che ci fanno respirare e per le loro fantastiche idee, ai capoeiristi lacustri, ai quali devo più di quello che pensano, a tutti quelli che lavorano per la diffusione del software e delle reti libere,

grazie.

Acknowledgements

To all my family, especially my mother and my father, for supporting me in all my choices, to Simona, for everything she does for me, to ninux.org, for making me grow, and especially to Nino, for dragging me into the project and pushing me to learn new things, to all the Tor Vergata Netgroup lab, especially to Saverio and Simone, to all my mates at Roma Tre, for all the good times we had together, to the guys of c-base and Freifunk, for all the things that they do, for the atmosphere that they make us breathe, and for their fantastic ideas, to the lake capoeiristas, to which I owe more than they might think, to the ones that work to spread free software and free networks,

thank you.

Introduzione

Il fenomeno delle *reti comunitarie* (*community networks*) si sta diffondendo, di recente, in tutto il mondo. Queste reti, spesso basate su tecnologie senza fili a basso costo e di solito nate dall'iniziativa di volontari tecnicamente preparati, si stanno evolvendo, in alcuni luoghi, in infrastrutture critiche per l'istruzione e le imprese.

Tipicamente, l'infrastruttura di una rete comunitaria ha molti proprietari e amministratori, in contrapposizione con il modello classico degli Internet Service Provider, in cui l'infrastruttura è posseduta e gestita da un singolo ente. La coordinazione tra gli amministratori di una rete comunitaria è minimale e talvolta limitata alla sola allocazione degli indirizzi IP, in maniera simile a ciò che avviene nella rete Internet a livello di Autonomous System.

Il *protocollo OLSR* (Optimized Link State Routing) per reti mobili adhoc (MANETs) è uno dei protocolli di routing più utilizzati dai membri delle reti comunitarie.

Crescendo e sviluppandosi, queste reti possono essere bersaglio di diversi tipi di attacchi, facendo emergere la necessità di meccanismi di protezione dell'infrastruttura. Qualsiasi scelta progettuale riguardante tali meccanismi deve tenere presente la mancanza di una forte coordinazione tra gli amministratori della rete, così come la natura eterogenea e decentralizzata delle reti comunitarie. Per questa ragione le soluzioni esistenti che fanno affidamento su un'amministrazione centralizzata, come le soluzioni basate su segreti condivisi tra i nodi della rete, devono essere scartate o modificate sostanzialmente.

In questo scenario, il traffico degli utenti potrebbe essere protetto da alcuni attacchi, come intercettazioni o furto di identità, mediante l'utilizzo di crittografia end-to-end, come quella fornita dal protocollo TLS. L'infrastruttura di routing, cioé i pacchetti relativi al protocollo di routing, devono, però, essere protetti in altri modi, ad esempio tramite l'utilizzo di crittografia a chiave pubblica e di sistemi di reputazione. Uno standard largamente accettato in entrambi questi campi è rappresentato da PGP (Pretty Good Privacy, da cui è stata derivata la specifica OpenPGP) e dal suo modello di *web of trust* (rete di fiducia). In tale modello, gli utenti PGP certificano l'identità di altri utenti PGP firmando crittograficamente le loro chiavi pubbliche. Quindi, basandosi su questo e su altre informazioni, quali la data di scadenza delle chiavi, un livello soggettivo di fiducia può essere calcolato da ogni utente PGP per ogni altro utente PGP, inclusi gli utenti sconosciuti. Dunque tra gli utenti si costruisce una "rete di fiducia": un sistema di reputazione decentralizzato, contrapposto alle infrastrutture a chiave pubblica (PKIs) gerarchiche, le quali si affidano ad entità centrali come le Certification Authority (CA).

Partendo da queste considerazioni e dalle soluzioni esistenti, in questa tesi è progettata e presentata la **Web of Trust OLSR Extension**, un'estensione del protocollo OLSR che mira a proteggere l'infrastruttura delle reti multi-amministrate, in particolare le reti comunitarie, da alcuni tipi di attacchi, quali furto di identità, immissione di false informazioni dall'esterno e attacchi di ripetizione, utilizzando firme digitali *hop-by-hop* nei pacchetti di controllo di OLSR, ed un appropriato meccanismo di scambio dei *timestamp*.[1]

Per di più, mentre il routing tradizionale basa le sue decisioni sul campo "destination" dei pacchetti IP in arrivo, alcuni sistemi operativi sono dotati di un meccanismo chiamato *policy based routing*, che permette di prendere decisioni di instradamento basandosi su altri campi dei pacchetti IP. Nel

[1] L'estensione non è basata su orologi sincronizzati, ma assume che la differenza tra gli orologi a bordo dei nodi rimanga, con una certa tolleranza, costante.

kernel Linux questa caratteristica è implementata dal Routing Policy Database (RPDB), che permette di specificare vari tipi di regole e l'utilizzo di diverse tabelle di instradamento.

La *Web of Trust OLSR Extension* utilizza il policy based routing per aggiornare parallelamente diverse tabelle di instradamento, ognuna corrispondente ad un diverso livello di fiducia, al momento della verifica della firma dei pacchetti OLSR ricevuti. La scelta della tabella di instradamento appropriata è basata sul livello di fiducia nei confronti del mittente del pacchetto. Livello di fiducia che l'amministratore del nodo calcola soggettivamente, utilizzando il modello del *web of trust*. Combinando queste tabelle con le regole appropriate, il traffico degli utenti può seguire percorsi corrispondenti ai diversi livelli di fiducia, delineando delle reti virtuali sovrapposte alla topologia reale.

La *Web of Trust OLSR Extension* è sviluppata come plug-in dell'implementazione, molto utilizzata nelle reti comunitarie, del protocollo OLSR dello UniK University Graduate Center di Oslo. Inoltre il plug-in è stato testato in un ambiente emulato.

Struttura della tesi

Nel primo capitolo viene presentato il protocollo OLSR ed alcune questioni relative alla sua sicurezza. Successivamente, nel capitolo 2 viene fornita una presentazione delle reti multi-amministrate, in particolare delle reti comunitarie, seguita dalla spiegazione relativa a PGP ed al suo modello di *web of trust* nel capitolo 3, e da quella relativa al *policy based routing* nel capitolo 4.

I capitoli 5 e 6 illustrano la *Web of Trust OLSR Extension*, l'estensione del protocollo OLSR progettata e sviluppata per questa tesi, inclusa la sua implementazione e la successiva fase di test.

Le considerazioni finali sul lavoro svolto e sul lavoro futuro, contenute nel capitolo 7, sono completate con un articolo, derivato da questa tesi e riportato nell'appendice A. Questo attualmente si trova in fase di elaborazione per essere inviato alla conferenza “IEEE International Conference on Communications 2009”.

La presente tesi è stata svolta esternamente presso il Dipartimento di Ingegneria Elettronica dell’Università degli Studi di Roma Tor Vergata.

Introduction

The *community networks* phenomenon is spreading, in recent years, all over the world. These networks, often based on low-cost wireless technologies and usually born from the initiative of technically skilled volunteers, are becoming, in some localities, critical infrastructures for education and business.

Typically, a community network infrastructure has several owners and administrators, as opposed to classic Internet Service Provider models, in which the infrastructure is owned and managed by a single organization. Coordination between community network administrators is minimal, and is sometimes limited to IP addresses allocation, in a way that resembles the Internet at the Autonomous System's level.

One of the most popular routing protocols employed in community networks is represented by the Optimized Link State Routing (OLSR) protocol for mobile ad hoc networks (MANETs).

As these networks grow and develop, they may become target of various threats, and thus the need for security emerges. Any design choice of a protection mechanism must take into account the lack of strong coordination between network administrators, as well as the heterogeneous and decentralized nature of community networks. For this reason existing solutions that rely on a centralized administration, such as solutions based on shared secrets between network nodes, must be discarded or substantially modified.

In this scenario, user traffic may be protected from some attacks, such as eavesdropping and identity spoofing, by using end-to-end encryption like the one provided by TLS, but the routing infrastructure, i.e. the routing protocol packets, must be secured by other means. Public key cryptography as well as reputation systems have to come into play. A widely accepted standard in both of these areas is represented by Pretty Good Privacy (PGP, from which the OpenPGP specification is derived) and its web of trust model. In this model, PGP users certify the identity of other PGP users by cryptographically signing their public keys. Then, based on this and other information, such as key expiration dates, a subjective level of trust may be computed by each PGP user for every other PGP user, including strangers. So a web of trust is created: a decentralized reputation system, as opposed to hierarchical Public Key Infrastructures (PKIs), that rely on central entities such as Certification Authorities (CAs).

Departing from these considerations, and from the existing solutions, the **Web of Trust OLSR Extension** is devised and presented in this thesis. It aims at protecting the networking infrastructure of multi-administered networks, especially community networks, from some threats, such as identity spoofing, link spoofing and replay attacks, by using hop-by-hop OpenPGP signatures in OLSR control packets, and an appropriate timestamp exchange mechanism[2].

Moreover, some operating systems provide a feature called *policy based routing*: the ability to take routing decisions on incoming traffic based on fields of the IP packets different from the usual "destination" field. In the Linux kernel this feature is implemented by the Routing Policy Database (RPDB), which permits the specification of multiple rules and the use of several routing tables.

The Web of Trust Extension uses policy based routing to update multiple routing tables, each corresponding to a different level of trust, upon the reception of an OLSR packet and the verification of its signature. The selection of the appropriate routing table is based on the subjective trust value

[2]The extension does not rely on synchronized time, but assumes that the time difference between network node's clocks remains constant, with some tolerance.

computed, using the web of trust model, by the receiving node's administrator with respect to the OLSR packet originator. Combining the routing tables with the appropriate rules, user traffic is able to follow network paths corresponding to different levels of trust, drawing virtual networks overlaid to the real network topology.

The extension is implemented as a plugin of the UniK OLSR Implementation (olsrd), popular among community network members. Subsequently some tests are performed in an emulated environment.

Structure of the Thesis

In chapter 1, an introduction to OLSR and related security issues will be presented. Then, in chapter 2, an overview of multi-administered networks, and especially community networks, is provided, followed by an explanation of PGP and its web of trust model, in chapter 3, and of policy based routing and the Linux RPDB in chapter 4.

In chapter 5 and chapter 6 the Web of Trust OLSR Extension, devised for this thesis, is illustrated, along with its implementation and testing.

The final considerations on past and future work in chapter 7 are completed with an article, derived from this thesis and reported in appendix A, that at the time of this writing is in the process of being completed and submitted for the IEEE International Conference on Communications 2009.

Contents

1 Securing the OLSR Protocol **1**

1.1 Optimized Link State Routing (OLSR) 1

1.1.1 Core Functionality 2

1.1.2 Auxiliary Functions 7

1.2 Security of OLSR 9

1.3 A Secure Extension to the OLSR Protocol 11

1.4 The UniK OLSR Implementation (olsrd) 15

1.5 Implementation of the OLSR security extension 16

2 Multi-administered networks **18**

2.1 Community Networks 18

2.2 Other scenarios 20

2.2.1 Airport MANET 20

2.2.2 Harbour MANET 22

3 Pretty Good Privacy (PGP) and the Web of Trust **23**

3.1 Pretty Good Privacy 23

3.1.1 Encryption and Decryption 24

3.1.2 Signature Creation and Verification 24

3.2 Web of Trust 26

3.3 GNU Privacy Guard (GnuPG) 27

3.3.1 GnuPG Made Easy (GPGME) Library 28

4 Policy Based Routing 29

4.1 Introduction to Policy Based Routing 29

4.2 Policy Based Routing in Linux 30

5 The OLSR Web of Trust Extension 34

5.1 OLSR Web Of Trust Extension 34

5.2 Security Considerations 43

6 Implementing and Testing the OLSR Web of Trust Extension 45

6.1 Implementation 45

6.2 Configuration 48

6.3 Testing 50

7 Conclusions, Current and Future Work 55

7.1 Conclusions 55

7.2 Current Work 56

7.3 Future Work 56

A Trusted Routing in Wireless Community Mesh Networks i

List of Figures

1.1 Basic OLSR Packet Format 2
1.2 MID Message Format 4
1.3 HELLO Message Format 5
1.4 TC Message Format 6
1.5 HNA Message Format 7
1.6 OLSR Attack Tree [1]. Non-leaf nodes represent goals, arrows departing from the same node represent different ways to achieve the same goal, and leafs represent the actual attacks. Gray leafs represent the attacks countered by the "Secure Extension to the OLSR Protocol" (§ section 1.3) and the "Web of Trust OLSR Extension", object of this thesis (§ chapter 5). 10
1.7 Secure OLSR basic signature message 11
1.8 Secure OLSR timestamp exchange challenge message 13
1.9 Secure OLSR timestamp exchange challenge-response message 14
1.10 Secure OLSR timestamp exchange response-response message 15
1.11 The Secure OLSR plugin design. 16

2.1 A multi-administered OLSR MANET in an airport. 21

3.1 PGP message encryption and decryption. 25
3.2 PGP signature creation and verification. 25

4.1 A map of Iptables chains functional to policy routing 32

5.1 Web of Trust OLSR extension basic signature message 37

5.2 Web of Trust OLSR extension timestamp exchange challenge message 37
5.3 Web of Trust OLSR extension timestamp exchange challenge-response message 38
5.4 Web of Trust OLSR extension timestamp exchange response-response message 38
5.5 A flow chart describing the algorithm used upon the arrival of a new signed packet. 39
5.6 A flow chart of the timestamp exchange algorithm. 40

6.1 The OLSR Web of Trust plugin design. 47

List of Tables

4.1 Summary of the selectors that may be specified by using the Iptables and Iproute tools. 33

6.1 New recommended values for some OLSR parameters. 49

6.2 Measured times for some cryptographic operations 54

Chapter 1

Securing the OLSR Protocol

In this chapter, the OLSR protocol for Mobile Ad Hoc Networks (MANETs) is introduced, along with some security related issues. Then, in section 1.3 the "Secure extension to the OLSR protocol" [2] is presented, which, using a shared secret to sign control packets, prevents some attacks to the routing infrastructure. This will be the basis for the "Web of Trust extension", described in section 5.1.

1.1 Optimized Link State Routing (OLSR)

The Optimized Link State Routing protocol (OLSR), described in IETF RFC 3626, is a popular networking protocol developed for mobile ad hoc networks. It is a *proactive* protocol and, as the name suggests, is based on *Link State* routing. OLSR does not rely on any central entity nor makes any assumption on the underlying link-layer protocol, is aware of asymmetric links but does not exploit them, and uses an optimization technique called *Multipoint Relaying (MPR)* to diffuse messages in the network.

RFC 3626 modularizes the protocol into a *core functionality* and a set of *auxiliary functions*.

1.1.1 Core Functionality

The core functionality is required for OLSR to operate and to provide routing in a stand-alone MANET.

Basic Packet Format

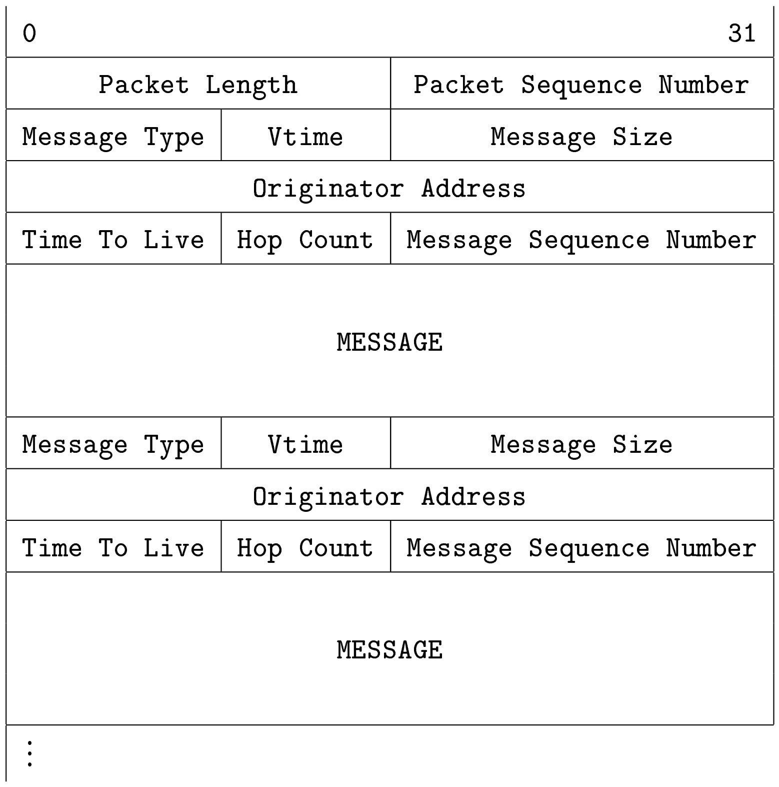


Figure 1.1: Basic OLSR Packet Format

OLSR packets are contained in UDP datagrams using IANA assigned port 698, and have the general format described in figure 1.1, where:

- **Packet Length** expresses the length of the packet, in bytes;
- **Packet Sequence Number** is incremented by one for each transmitted packet and is maintained separately for each OLSR interface[1].

[1]An *OLSR interface* is a network interface participating in an OLSR MANET.

OLSR messages, which may be of various types, are contained in the rest of the packet. All message types share a common header, with the following fields:

- **Message Type** specifies the type of the message;
- **Vtime** indicates the length of the validity time regarding the information contained in the message;
- **Message Size** specifies the size of the message;
- **Originator Address** is the *Main Address*[2] of the node that generated the message;
- **Time To Live** contains the maximum number of times that a message must be forwarded over the network, and is decremented at every hop;
- **Hop Count** is incremented at every hop;
- **Message Sequence Number** is a number that is unique for each message and is assigned by the originator node.

Multiple Interface Declaration (MID) Messages

When a node has multiple *OLSR interfaces* participating in an OLSR MANET, the association between its *Main Address* (see note 2 on page 3) and the addresses of all its OLSR interfaces must be announced to the other nodes in the network. This is accomplished through Multiple Interface Declaration (MID) messages, whose format is shown in figure 1.2.

The **OLSR Interface Address** field contains the address of an OLSR interface associated to the *Main Address* indicated in the **Originator Address** field.

[2]The *Main Address* is an IP address that a node must choose as its *node id* among all the available addresses on all OLSR interfaces.

0		31
MID_MESSAGE	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
OLSR Interface Address		
OLSR Interface Address		
⋮		

Figure 1.2: MID Message Format

Multipoint Relays (MPR)

Classical flooding (i.e. forwarding of received messages to all neighbors) is very expensive in terms of bandwidth. That's why OLSR uses *Multipoint Relaying*: an optimization obtained by avoiding redundant retransmissions.

To achieve this, each node selects among its symmetric 1-hop neighborhood[3], considering their willingness[4], some nodes as *Multipoint Relays (MPRs)*, so that each 2-hop neighbor[5] can be reached through an MPR. The set of MPRs selected by node X is called the *MPR set* of X. Nodes that are in the MPR set of other nodes have the responsibility of forwarding their messages.

HELLO Messages

HELLO messages are used for the purpose of link sensing, neighborhood detection and MPR selection. Emitted at fixed time intervals, contain information about the status of the node's links and neighbors. Their format is shown in figure 1.3.

- **HTime** specifies the size of the time interval between emissions of subsequent HELLO messages;

[3]The *symmetric 1-hop neighborhood* of a node is the set of nodes which have at least one symmetric link with the node itself.

[4]Willingness is explained at page 6.

[5]A *2-hop neighbor* is a node heard by a neighbor.

0			31
HELLO_MESSAGE	Vtime	Message Size	
Originator Address			
Time To Live	Hop Count	Message Sequence Number	
Reserved		Htime	Willingness
Link Code	Reserved	Link Message Size	
Neighbor Interface Address			
Neighbor Interface Address			
...			
Link Code	Reserved	Link Message Size	
Neighbor Interface Address			
Neighbor Interface Address			
...			
⋮			

Figure 1.3: HELLO Message Format

- **Willingness** indicates, with a number between 0 (`WILL_NEVER`) and 7 (`WILL_ALWAYS`), the willingness of the node to carry and forward traffic for other nodes;
- **Link Code** specifies information about the link (asymmetric, symmetric, lost, ...) and neighbor (symmetric neighbor, MPR, ...) whose interface is indicated in the associated **Neighbor Interface Address** fields;
- **Link Message Size** is the distance, in bytes, measured between two subsequent **Link Code** fields (or the end of the message).

Topology Control (TC) Messages

In *Link State* routing, each node spreads information about its neighbors over the whole network. In OLSR this task is achieved by *Topology Control (TC)* messages, whose format is shown in figure 1.4.

0		31
TC_MESSAGE	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
ANSN		Reserved
Advertised Neighbor Main Address		
Advertised Neighbor Main Address		
...		

Figure 1.4: TC Message Format

- **Advertised Neighbor Sequence Number (ANSN)** is incremented every time the node's neighbor set changes;
- **Advertised Neighbor Main Address** contains the main address of a neighbor node.

If TC Redundancy (see subsection 1.1.2, page 8) is not used, TC messages are emitted by MPR nodes only.

1.1.2 Auxiliary Functions

The purpose of the auxiliary functions of OLSR is to add functionalities that may be applicable to specific scenarios. Their separation from the core functionality aims at keeping the protocol as simple as possible while adding complexity only when needed.

Host and Network Association (HNA) Messages

When a node has some network interfaces participating in an OLSR MANET and other interfaces which do not, it may be desirable to inject routing information in OLSR. *Host and Network Association (HNA)* messages serve for this task. Their format is displayed in figure 1.5.

0		31
HNA_MESSAGE	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Network Address		
Netmask		
Network Address		
Netmask		
...		

Figure 1.5: HNA Message Format

The **Network Address** and **Netmask** pair of fields specify the non-OLSR networks' data to be injected inside an OLSR MANET.

Link Hysteresis

To prevent unstable links (not rare over the wireless medium) from having consequences on the stability of the information maintained by OLSR nodes or even affect the routing process, the *Link Hysteresis* mechanism may be applied.

It uses an upper and a lower threshold on the link quality. Asymmetric links may be considered by the protocol as symmetric only when their link quality is greater than the upper threshold, while symmetric links may be considered asymmetric only when their link quality is less than the lower threshold. In this way a delay is introduced in the link sensing process, but greater stability is achieved.

TC Redundancy

By using the *TC Redundancy* mechanism, the robustness of the network topology information is increased. When an OLSR node parameter, called `TC_REDUNDANCY`:

- is 0, then the node includes in TC messages only the links with its *MPR selector set*[6];
- is 1, then the node includes in TC messages the links with its *MPR selector set* and with its *MPR set*;
- is 2, then the node includes in TC messages all of its symmetric neighbors.

MPR Redundancy

Optimization achieved by Multipoint Relaying may be traded off with more robustness with respect to topology changes by increasing the number of selected MPRs per node. This can be useful, for example, in mobile environments, where may be desirable to have the reachability of a node advertised by more nodes. A parameter called MPR_COVERAGE affects the

[6]The *MPR selector set* of a node is constituted by the nodes that selected it as MPR.

MPR selection process by specifying through how many MPRs every two-hop neighbor should, if possible, be reached.

1.2 Security of OLSR

This section focuses on the security of the OLSR protocol control messages, without taking into consideration user traffic, which may be protected and authenticated by other means (for example by employing TLS [3]).

In RFC 3636 no security measures are specified for OLSR. However, some security considerations are made on:

- **confidentiality**: to protect topological information from eavesdropping, PGP or symmetric encryption may be performed on control traffic;
- **integrity**: message authentication is recommended, in order to avoid injection of invalid control traffic.

In [4] and [1], the security of OLSR infrastructures is further explored. Vulnerabilities of proactive routing protocols are represented in an attack tree [5] similar to the one in figure 1.6. Some of these attacks are:

- **jamming**: in a wireless ad-hoc network a node may generate a massive amount of interfering radio transmissions. This cannot be prevented at the routing protocol level;
- **incorrect control traffic generation**: by e.g. identity spoofing or link spoofing;
- **incorrect control traffic relaying**: if control messages are not properly relayed, connectivity losses may result.

The "Web of Trust Extension" described in section 5.1 aims at countering the attacks denoted by a gray background in the OLSR attack tree reported in figure 1.6, i.e. replay attacks and generation of incorrect control messages. More security considerations are included in section 5.2.

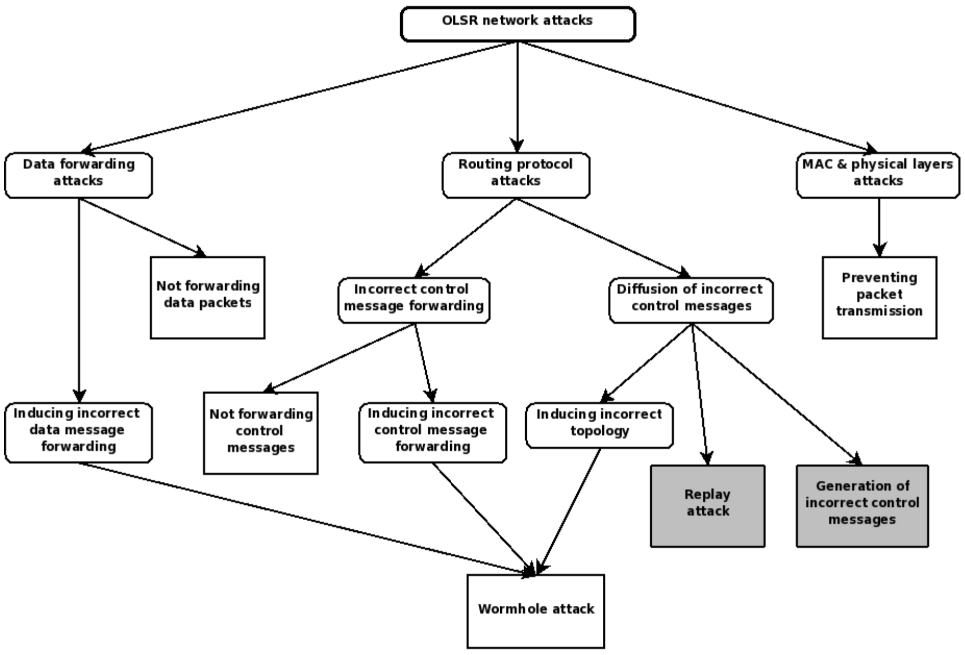

OLSR network attacks
Data forwarding attacks
Routing protocol attacks
MAC & physical layers attacks
Not forwarding data packets
Incorrect control message forwarding
Diffusion of incorrect control messages
Preventing packet transmission
Inducing incorrect data message forwarding
Not forwarding control messages
Inducing incorrect control message forwarding
Inducing incorrect topology
Replay attack
Generation of incorrect control messages
Wormhole attack

Figure 1.6: OLSR Attack Tree [1]. Non-leaf nodes represent goals, arrows departing from the same node represent different ways to achieve the same goal, and leafs represent the actual attacks. Gray leafs represent the attacks countered by the "Secure Extension to the OLSR Protocol" (§ section 1.3) and the "Web of Trust OLSR Extension", object of this thesis (§ chapter 5).

1.3 A Secure Extension to the OLSR Protocol

The “Web of Trust OLSR Extension” described in section 5.1 is based on the “Secure Extension to the OLSR Protocol” by Hafslund, Tønnesen et al., presented in [2] and implemented as an olsrd plugin; hence is here summarized. It should be noticed that by using a shared secret, this extension is not suitable for multi-administered networks with weak coordination between node administrators (§ chapter 2).

The extension described in [2] provides integrity for OLSR control messages through digital signatures obtained using a *symmetric key* shared between the nodes of an OLSR MANET. No confidentiality or integrity for user traffic is provided.

Control traffic is signed by every forwarding hop, thus the assumption made is that each forwarder trusts the previous forwarder and the source of the message[7].

To prevent replay attacks, timestamps are used and a timestamp exchange mechanism is introduced. Time between nodes is not assumed to be synchronized, but *relatively synchronized* i.e. running with relatively equal frequency.

0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Scheme	Algorithms	Reserved
Timestamp		
Signature (160 bits)		

Figure 1.7: Secure OLSR basic signature message

OLSR is extended with four new message types. Figure 1.7 shows the

[7]This assumption is straightforward, as all nodes share a common secret.

basic signature message, where:.

- **Scheme** - specifies the scheme used for the signature;
- **Algorithms** - specifies the algorithms used for the signature;
- **Timestamp** - is used to prevent replay attacks;
- **Signature** - is obtained as the hash of *all* the fields of the *OLSR packet* that precede the signature field itself plus the shared key, i.e.
  - the OLSR packet header,
  - all OLSR messages in the packet except the signature message,
  - the header of the signature message, the sub header and the timestamp,
  - the shared key.

The basic signature message must be the last message in the packet.

Message sequence numbers used in OLSR messages are not enough to prevent replay attacks, as:

- with 16 bits wrap-around is too frequent;
- an attacker could record one-hop traffic sent by a node and play it back in an area where it has never been heard.

Therefore 32-bit timestamps are used and exchanged upon connection through a three-way handshake made of a *challenge message* (figure 1.8), a *challenge-response message* (figure 1.9), and a *response-response* message (figure 1.10).

The three messages are self-signed in order to separate the validation of the signature message from the timestamp exchange mechanism, which is likely to take place between neighbors that have no registered timestamp of each other. The handshake between nodes A and B can be schematized in this way:

1. challenge: $A \rightarrow B : Ch_a D(M, K)$;

0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Destination		
Random value "challenge"		
Signature (160 bits)		

Figure 1.8: Secure OLSR timestamp exchange challenge message

2. challenge-response: $B \to A : Ch_b Ts_b D(IP_b, Ch_a, K) D(M, K)$;

3. response-response: $A \to B : Ts_a D(IP_a, Ch_b, K) D(M, K)$;

where:

- Ch_x is a 32-bit nonce generated by node X;
- $D(d_1, d_2, \ldots)$ the digest of the concatenation of $d_1, d2, \ldots$;
- Ts_x the timestamp of node X;
- IP_x the main address of node X;
- M the entire message;
- K the shared key.

Hence, the timestamp exchange begins with node A sending a digitally signed challenge message containing a 32-bit nonce. B receives the challenge message from A, generates the digest of its IP address, the received nonce and the shared key, sends a new 32-bit nonce along with its timestamp and digitally signs and sends the challenge-response message. When A receives the challenge-response message from B, tries to verify the data; if it fails discards the message, else computes the difference between its timestamp and B's and records it. Then A generates a response-response message with

0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Destination		
Random value "challenge"		
Timestamp		
Response Signature (160 bits)		
Signature (160 bits)		

Figure 1.9: Secure OLSR timestamp exchange challenge-response message

its timestamp, a digest of its IP address, a digital signature, and sends it to B. B verifies the data sent by A, and if it succeeds, uses the timestamp received to calculate the difference with its timestamp and record it.

After the timestamp exchange, difference between timestamps is calculated for every received signature message. If the recorded difference for the sender is T_N, a received signature message is valid if the difference T_0 between the local timestamp T_L and the timestamp T_R received in the message (i.e. $T_0 = T_L - T_R$), verifies $T_0 = T_N + S$ or $T_0 = T_N - S$, where S is a certain allowed slack.

In order to avoid a *Denial of Service (DoS)* attack, obtained by sending a big number of challenge messages with the aim of overloading the network and the processing units of the nodes, a timer is used. When a challenge message originating from a node X is received, a timer for X is set, and while the timer has not timed out, challenge messages originating from X are discarded before verification.

0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Destination		
Timestamp		
Response Signature (160 bits)		
Signature (160 bits)		

Figure 1.10: Secure OLSR timestamp exchange response-response message

1.4 The UniK OLSR Implementation (olsrd)

Many RFC 3626 compliant implementations of the OLSR protocol exist. For this thesis the *UniK OLSR Implementation*, also known as *olsrd*, initially developed by Andreas Tønnesen at the UniK University Graduate Center of the University of Oslo, and described in [6], was chosen. The choice was determined by this implementation's:

- **modularity** - new plugins can be added without changing the main codebase;
- **license** - the source code is released under a BSD-style license[8] [8], i.e. it is freely available and modifiable;
- **popularity** - it is used by many *community networks*[9], who also actively contribute to test, extend and enhance it;
- **portability** - written in pure C, runs on a wide range of hardware platforms (i386, ARM, MIPSEL, ...), including embedded devices, and

[8]Originally olsrd was released under the GNU Public License [7].

[9]See section 2.1, page 18.

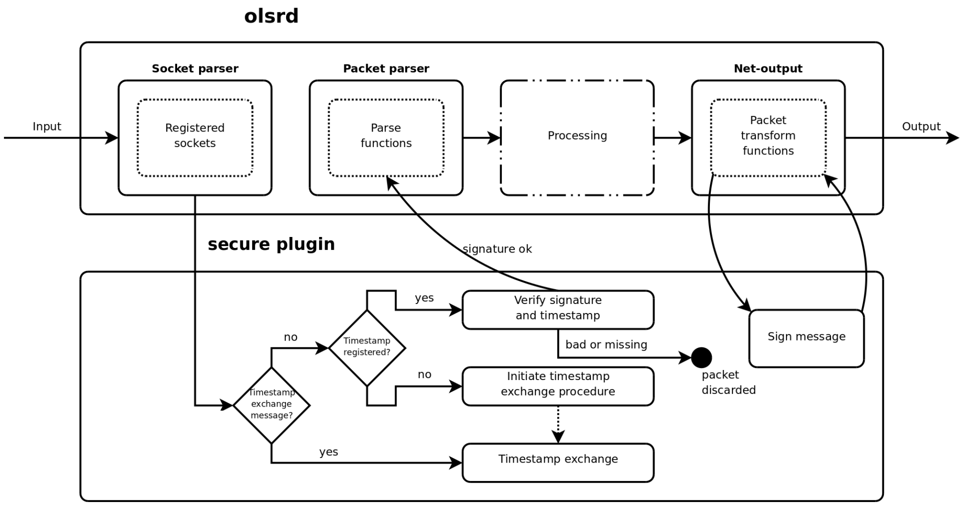


Figure 1.11: The Secure OLSR plugin design.

operating systems (GNU/Linux, Mac OS, OpenWRT, *BSD, Windows, . . .).

- **performance**: very low CPU time usage and high scalability.

Olsrd is maintained at the *olsr.org* website and has evolved into the *OLSR-NG* [10] project, aimed at making "olsrd into a rock solid product and a great routing daemon with high scalability"[10].

1.5 Implementation of the OLSR security extension

The Secure Extension to the OLSR protocol [2] is implemented as an olsrd plugin, using the SHA-1 hashing algorithm[11] [12], or MD5[12] [13].

The plugin intercepts incoming traffic in order to verify eventual signatures. Timestamp exchange messages are processed before signature messages. After successful verification of a signature message, this is removed,

[10]http://www.olsr.org/?q=background
[11]Through the OpenSSL library [11].
[12]Included, no dependencies needed, thus suited for embedded devices.

the Packet Length field is updated[13], and the packed is passed on to the other olsrd components for further processing. If signature verification fails the packet is discarded.

Records of registered timestamps are maintained by each Secure OLSR node. When a host receives a signature message from a node for which has no registered timestamp, the timestamp-exchange process is initiated.

[13]The signature's Message Size is subtracted from the Packet Length.

Chapter 2

Multi-administered networks

This chapter describes, in order to better understand the problems that this work addresses, the context in which the OLSR Web of Trust extension, described in section 5.1 and object of this thesis finds its application: multi-administered networks, with weak coordination between administrators. The most concrete example of such networks is represented by *community networks*, introduced below.

2.1 Community Networks

Since the first years of 2000, the social phenomenon[1] of *community networks* has emerged all over the world. These grassroots networks, initially built spontaneously by skilled volunteers, have become critical infrastructures for some business models [15, 16] and education [17].

One of the characteristic features of community networks is that the infrastructure is owned and managed (tough owner and administrator often coincide) by several entities, which act both as end users and service providers, as opposed to traditional service providers, who exclusively own and manage their networking infrastructures. The motivations that drive the phenomenon of community networks are various:

[1]Born from the Seattle Wireless Idea [14].

- **low cost of hardware**, most notably of IEEE 802.11a/b/g compliant devices. Moreover some components (such as antennas), may be self-constructed from recycled material (e.g. coffee cans);
- **low cost of deployment**, especially if wireless technologies are employed;
- **narrow the digital divide**: some areas, such as rural or mountainous, are not profitable for traditional ISPs, because the morphology of the terrain and the population density lead to high deployment and maintainment costs in the last mile, not counterbalanced by a (quantitatively) low demand for Internet connectivity [18] [19, 17, 20, 21];
- **learn, experiment, have fun and share knowledge**, sometimes in collaboration with universities or enterprises, in a way similar to what happens for FLOSS (Free/Libre/Open Source Software), which is also frequently used to run and manage community networks [22, 23, 14, 21, 24];
- **political or social reasons**: bypass the traditional ISPs to avoid censorship, logging; and tracking [25] or simply to achieve horizontal, nonhierarchical communication.

Even if any available technology is used for the deployment of these networks, wireless links are prevalent, and so the expression “wireless community network” is often employed. For the same reason the routing protocols used in these networks have to be suitable for the wireless medium, or, better, to mixed mediums, like OLSR, AODV [26], B.A.T.M.A.N. [27] or BGP [28, 21].

Usually, community network members provide, along with Internet access, local (intranet) services, such as voice over IP, DNS, gaming, file sharing, Web pages, weather stations, to other community members.

Community networks focused on experimentation and knowledge sharing have developed their own solutions, by modifying Linux distributions [29, 30], extending[2] or developing brand new protocols [27, 32] and helping the development of network card drivers [33].

In order to regulate data transit between nodes, some agreements, analogous to FLOSS licenses, such as the Pico Peering [34] and Freenetworks.org [35], have been devised by community network members.

Even if these kind of networks have, were deployed, advantages for citizens and local enterprises, the regulatory environment does not always help their development [18].

Also metropolitan networks constituted by hot-spots for Internet connection sharing are sometimes identified as community networks [36, 37], but are beyond the scope of this work.

2.2 Other scenarios

In addition to community networks, in this section some other application scenarios for the *Web of Trust OLSR Extension* presented in chapter 5 are depicted.

2.2.1 Airport MANET

In the following scenario, set in an airport, a networking infrastructure is built between airplanes on the ground and the airport's network. Each airplane, from same or different flight companies, incorporates a wireless OLSR node. In this way, each landed airplane may act both as user of the network, by sending and receiving data, and as service provider, by forwarding other airplane's traffic. Other nodes are installed in the airport's facilities and others may be installed on board of the ground vehicles used for baggage or passenger transport.

Such network could be the target of several types of attacks (§ section 1.2), and thus the networking infrastructure should be secured. Using

[2]Some experimental extensions to the OLSR protocol, such as *Link Quality*[31] and *Fish-Eye routing* have been designed and implemented in olsrd [9] by community network members.

a shared secret solution would require strong coordination and trust between node administrators, which is not always the case.

In this scenario, if the nodes use the OLSR protocol, the extension described in section 5.1 could be employed in order to obtain a secured infrastructure, without the need for strong central coordination and full trust between network administrators.

This interconnection may then be exploited in order to give low-cost, best-effort Internet access to passengers waiting in the airplane during an intermediate stop, or to exchange non-critical data[3] with the flight companies' offices before a new take-off.

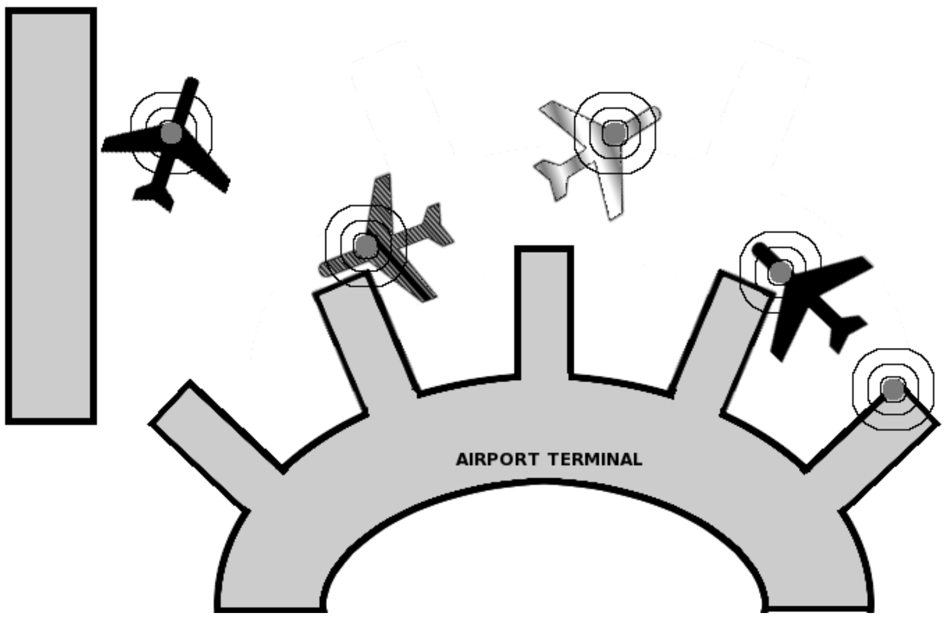


Figure 2.1: A multi-administered OLSR MANET in an airport.

[3]Due to the mobility of the nodes and the unreliability of the wireless medium, is very difficult to guarantee a minimal quality of service.

2.2.2 Harbour MANET

A scenario similar to the one depicted in the previous section could be transposed in an harbour, where boats could create a low-cost OLSR MANET among themselves. This network could then be used to connect to local services provided by the harbour's authorities or to provide Internet access on board of the boats.

Chapter 3

Pretty Good Privacy (PGP) and the Web of Trust

This chapter is focused on the *Pretty Good Privacy (PGP)* cryptosystem and its *Web of Trust* model, used for the validation of the ownership of cryptographic keys.

3.1 Pretty Good Privacy

Pretty Good Privacy (PGP) is the name of the cryptosystem and open source computer program originally developed by Phil Zimmermann in 1991, with the aim of protecting users' privacy when using electronic communication technologies (especially e-mail). The success of PGP brought to the definition of OpenPGP, an Internet Engineering Task Force (IETF) Proposed Standard, specified in [38].

PGP combines symmetric and asymmetric cryptography in order to obtain high levels of security at reasonable speeds. Supported operations include encryption of text or binary data, decryption, as well as digital signature creation and verification.

3.1.1 Encryption and Decryption

When encrypting a plaintext, at first compression is applied, in order to reduce data size and increase cryptographic strength by removing repeated patterns, which may be exploited by some cryptanalysis techniques. Then a random session key is generated, using entropy gathered from user-generated events, such as mouse movements. This one-time session key is at first used to encrypt the compressed plaintext using a symmetric and fast compression algorithm, and then is in its turn encrypted using the recipient's public key. Finally the obtained cyphertext and encrypted session key are transmitted, eventually over an unsecured medium, to the recipient.

In order to decrypt, the session key used by the sender is extracted using the recipient's private key. Then the original plaintext can be obtained by applying subsequently the same symmetric algorithm used by the sender and decompression.

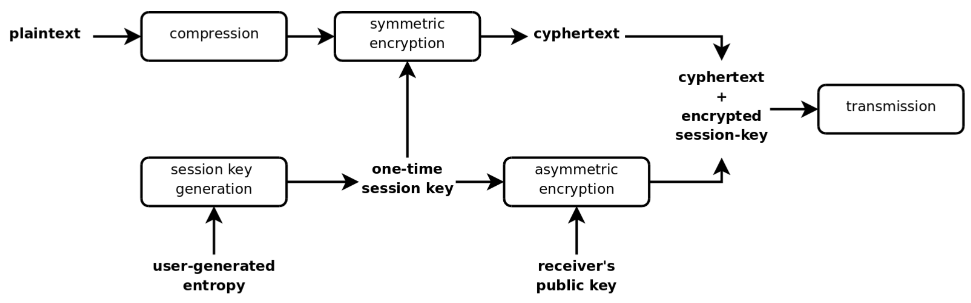
A diagram that describes PGP message encryption and decryption can be found in figure 3.1.

3.1.2 Signature Creation and Verification

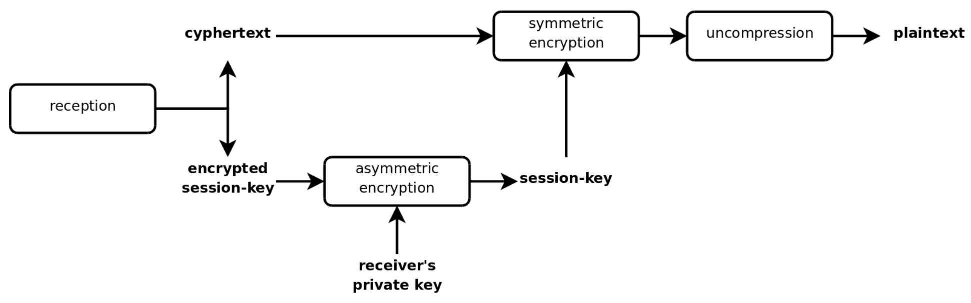
Signature is obtained by applying a one-way cryptographic hash to the plaintext to be signed, thus obtaining a fixed-length message digest that is encrypted using the sender's private key. The result of these operations is the signature, that is sent with the message.

The recipient can verify a received signature by decrypting it using the sender's public key, obtaining the message digest, and then comparing it with an hash computed by herself.

Signature creation and verification operations are schematized by figure 3.2.

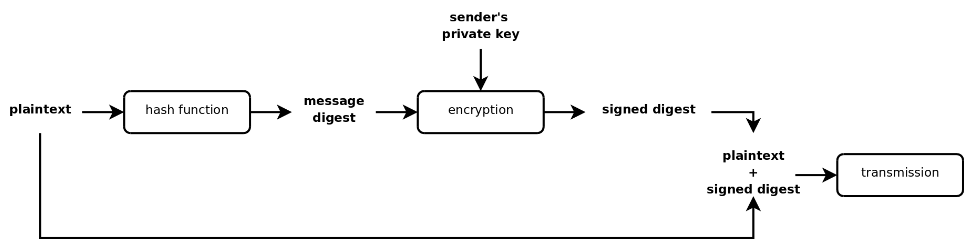


(a) message encryption by the sender

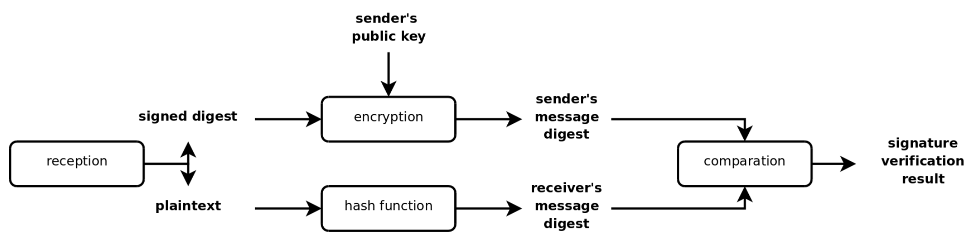


(b) message decryption by the receiver

Figure 3.1: PGP message encryption and decryption.



(a) signature creation by the sender



(b) signature verification by the receiver

Figure 3.2: PGP signature creation and verification.

3.2 Web of Trust

PGP uses a distributed, non-hierarchical trust model called *Web of Trust* to validate public keys' ownership.

Traditional, hierarchical **Public Key Infrastructures (PKIs)** rely on a root Certification Authority (CA), trusted by all users, who, using signed electronic documents called Digital Certificates, assures the validity of public keys. A CA may act as an *introducer*, i.e. validate public keys directly, or as a *meta-introducer*, i.e. empower lower level Certification Authorities to validate keys in its place.

CAs are also responsible for issuing Certificate Revocation Lists (CRLs), which include certificates whose validity has been revoked.

The whole structure relies on all users' trust in the root CA and on the secrecy of the root CA's private key.

By contrast, in the **Web of Trust** model [39], each user validates other users' public keys by signing them, and thus becoming an *introducer* of these. When key signature takes place, a *check level* and an *owner trust* level are assigned to the signed public key.

The check level specifies how carefully the signer has checked the identity of the key owner, and its value can range from 0 to 3:

- 0 means that no particular claim is made from the signer on the carefulness of identity verification;
- 1 means that the identity of the signed key owner was not verified, but the signer believes that the identity is correct;
- 2 means that identity verification was done;
- 3 means that very careful identity verification was done.

The owner trust level specifies the trust the signing user puts on the signed key owner's to correctly certify, by signing, other keys, and usually is not published. Four owner trust values are allowed:

- Ultimate or Implicit trust;
- Complete trust;
- Marginal trust;
- Untrusted or No trust.

The *validity level* is a subjective indicator of how much an user considers a public signed key to be valid. It is *computed* from the affixed signatures, the check level and the (local) owner trust, and may assume three different values:

- Invalid;
- Marginally valid;
- Valid.

Using the default model, a key is valid if, in terms of owner trust, a completely trusted signature or three marginally trusted signatures are affixed to it.

Thus users trust other users to certify key validity, without the need for a central Certification Authority, creating a web of trust among themselves: a decentralized and fault-tolerant reputation system.

3.3 GNU Privacy Guard (GnuPG)

GNU Privacy Guard (GnuPG) [40] is the GNU project's [41] full implementation of the OpenPGP [38] specification. Born in 1997 and developed mainly in Europe due to United States' export regulations on cryptography[1],

[1]The same US export regulations for which Phil Zimmermann was sued in 1993: cryptosystems using keys longer than 40 bits were considered weapons. The author notes that the decentralized and grass-roots nature of the Web of Trust model, paired with the history and political considerations that brought PGP into existence [42], well couples with the principles and ideas that inspire several community networks.

it supports encryption, decryption, signature creation and verification, and complete key management. Other features include:

- better functionality than PGP and some security enhancements over PGP 2;
- decryption and verification of PGP 5, 6 and 7 messages;
- supports ElGamal, DSA, RSA, AES, 3DES, Blowfish, Twofish, CAST5, MD5, SHA-1, RIPE-MD-160 and TIGER algorithms;
- easy implementation of new algorithms using extension modules;
- the User ID is forced to be in a standard format;
- supports key and signature expiration dates;
- multi-language support;
- online help system;
- supported platforms: GNU/Linux (x86, Alpha, MIPS, SPARC64, M68k or PowerPC architectures), FreeBSD, OpenBSD, NetBSD, Windows 95/98/NT/2000/ME/XP, MacOS X, ...;
- integrated support for HKP keyservers [43].

3.3.1 GnuPG Made Easy (GPGME) Library

GnuPG Made Easy (GPGME) is a C language library that provides an Application Programming Interface (API) for GnuPG message encryption, digital signature and key management operations. It is released under the GNU Public License and is freely available at [40].

GNU Privacy Guard and GnuPG Made Easy have been employed in the implementation of the Web of Trust OLSR plugin described in 5.1.

Chapter 4

Policy Based Routing

This chapter introduces *policy based routing* (or simply *policy routing*): the ability, belonging to some operating systems, to take routing decisions on incoming IP traffic based on information other than the IP "Destination address" field. This, coupled with complex rules and multiple routing tables, allows for the creation of overlay networks over actual network topologies, such as the ones built by the Web of Trust OLSR Extension, described in chapter 5 and object of this thesis.

Mostly the policy based routing features of GNU/Linux [44] will be taken into account, as it is the most widespread operating system supporting this feature, and for the reason that many of the tools used to implement the OLSR Web of Trust extension are mainly designed for GNU/Linux operating systems.

4.1 Introduction to Policy Based Routing

Conventional IP routing can be described as "a destination-driven process" [45], with best-effort packet delivery. In other words, at each forwarding router, the routing decision is entirely based on the IP "Destination address" field, and no guarantee is given on data delivery, nor on bandwidth.

With the growth of the Internet, the need has emerged for more sophisticated traffic engineering tools, able to address and shape data streams, in order to achieve guaranteed quality of service.

By using policy based routing, a forwarding router can decide a packet's next hop by examining the entire IP packet, especially all the fields in the IP header.

In this way routing can be based, for example, on the "Source address", "Type of Service" or "Protocol" fields, providing different paths on the network for different users or classes of traffic.

Usually, policy routing is enforced by a set of rules and/or multiple routing tables.

4.2 Policy Based Routing in Linux

Policy routing in the Linux kernel is implemented by the *Routing Policy Database* (RPDB), which contains three different types of elements: addresses, routes and rules. By combining these elements between them and using multiple routing tables, several policies may be enforced.

A collection of utilities called *Iproute2* [46] provides easy access to the RPDB. Using the *ip* tool, entries may be added to different routing tables, which are numbered from 1 to 255. Some of these tables have a special meaning:

- table **255** is table **local**. Maintained automatically by the kernel, contains routes for local and broadcast addresses;
- table **254** is table **main**. This corresponds to the "traditional" routing table;
- table **253** is table **default**. Reserved for post-processing, if no matches are found for the current packet in table 255 or table 254.

Moreover, *rules* may be defined, with associated priority values, in order to affect the routing decision process. Up to 2^{32} different rules may be specified, each with a selector and an action predicate.

When a packet is processed, the rules are scanned in order of increasing priority. If the packet matches a selector, then the corresponding action is performed.

The selector may match IP packet fields such as source address, destination address, TOS, or other information such as the incoming interface. Moreover, in combination, the *Iptables/Netfilter* [47] tool may be employed in order to mark packets using very powerful selection criteria. The marked packets may then be matched by a rule's selector using the "fwmark" specifier. This packet marking is performed in the "mangle-prerouting" table (§ figure 4.1). Most of the policies that may be expressed in such fashion are summarized in table 4.1. For further information, the Iproute and Iptables documentation may be referenced.

Actions associated to rules may be of one of the following types:

- **unicast**. Return the route found in the routing table referenced by the rule;
- **blackhole**. Silently drop the packet;
- **unreachable**. Generate a "Network is unreachable error";
- **prohibit**. Generate a "Communication is administratively prohibited" error;
- **nat**. Translate the source address of the IP packet into some other value.

Some rules are pre-defined in the RPDB:

- with priority 0, match all packets, lookup routing table local (255). This rule cannot be overridden;
- with priority 32766, match all packets, lookup routing table main (254);
- with priority 32767, match all packets, lookup routing table default (253);

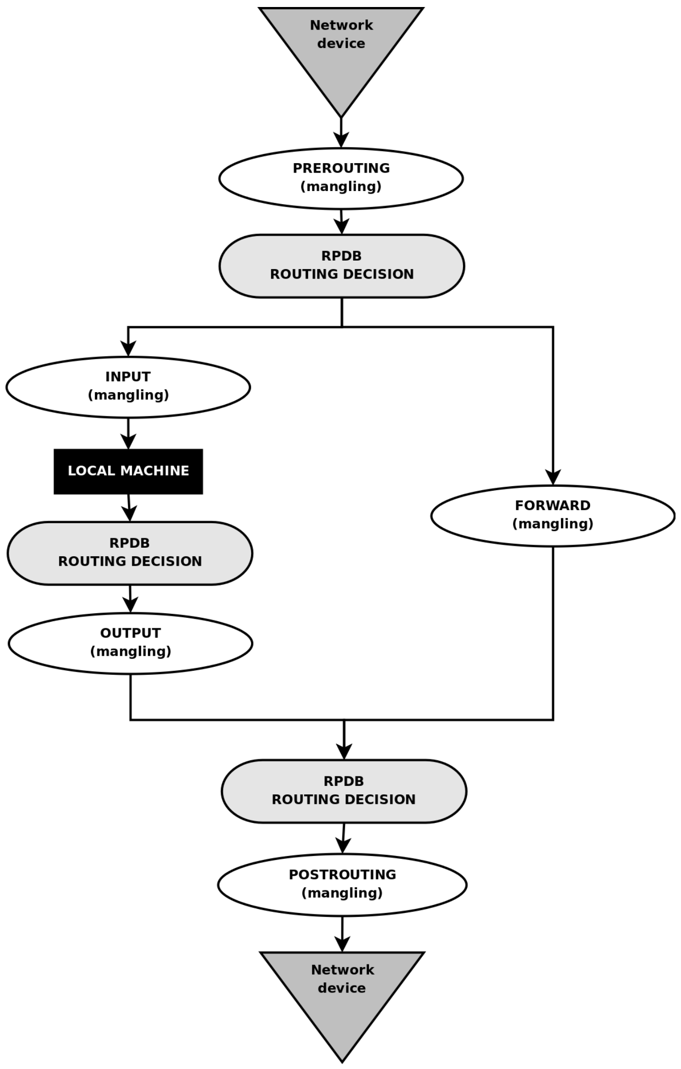


Figure 4.1: A map of Iptables chains functional to policy routing

Selector	Commands
destination address/network	ip rule
source address/network	ip rule
tos	ip rule
interface	ip rule dev
packet length	iptables + ip rule fwmark
ip header	iptables u32 + ip rule fwmark
destination port	iptables + ip rule fwmark
source port	iptables + ip rule fwmark
udp header	iptables u32 + ip rule fwmark
tcp header	iptables + ip rule fwmark
address type (unicast,broadcast,...)	iptables + ip rule fwmark
packet payload	iptables string + ip rule fwmark
mac address	iptables + ip rule fwmark
ipsec spi	iptables + ip rule fwmark
ipsec packet handling policy	iptables + ip rule fwmark
transmitted packets or bytes	iptables + ip rule fwmark
average packets or bytes	iptables + ip rule fwmark
number of parallel connections from same client	iptables + ip rule fwmark
dscp - tos	iptables + ip rule fwmark
owner process	iptables + ip rule fwmark
realm	iptables + ip rule fwmark
random	iptables statistic + ip rule fwmark
packet arrival time	iptables time + ip rule fwmark
raw bytes	iptables u32 + ip rule fwmark

Table 4.1: Summary of the selectors that may be specified by using the Iptables and Iproute tools.

Chapter 5

The OLSR Web of Trust Extension

The Web of Trust OLSR extension, object of this thesis, aims at bearing the PGP Web of Trust reputation system, described in section 3.2, into multi-administered OLSR networks, introduced in chapter 2. The extension is modelled on the *Secure Extension to the OLSR Protocol* [2] summarized in section 1.3, but due to the different objectives and nature of the two extensions, and to variable-size signatures, some fundamental modifications are introduced.

5.1 OLSR Web Of Trust Extension

The distributed design of MANETs make them suitable to scenarios in which the administration (or even ownership) of the network is decentralized[1]. Some OLSR extensions, including the above-mentioned, *Secure Extension to the OLSR Protocol* [2], based on a shared key, assume the management of the network to be centralized or, at least, strongly coordinated.

The *Web of Trust OLSR Extension* has the objective of building a trusted networking infrastructure minimizing at the same time the need

[1]See chapter 2 on multi-administered networks.

for central entities, and to permit the creation of multi-level trusted paths by populating different routing tables[2] associated to different levels of trust.

A shared key mechanism is not suitable for this task, so asymmetric key cryptography has to come into play. A widely tested and accepted standard in this area is represented by OpenPGP[3] [38].

Moreover, the users of OpenPGP implementations, such as GNU Privacy Guard (GnuPG), create a *Web of Trust* among themselves by signing and distributing each other's public keys. Each user computes a subjective *trust value* for every other user, based on trust rankings given by others or by herself. These trust values are used by each node using the Web of Trust extension to create entries in different routing tables, each associated to a trust level.

The extension is focused on the infrastructure, and does not provide confidentiality or integrity for user data. Only the packets of the routing protocol are signed, hence authenticated, and verified for integrity.

The signature message and the timestamp exchange messages described in the existing OLSR secure extension [2] (§ section 1.3), designed for symmetric cryptography and fixed size signatures, are not suitable for variable size signatures, or signatures whose size is not fixed at a constant value, such as OpenPGP signatures [38]. However, departing from these messages, a new secure extension, called "Web of Trust" extension, is here devised.

First of all, it should be considered that OLSR messages must be aligned on 32 bits, so if variable size signatures are used, a variable number of padding bits have to be appended at the end of the messages, until 32-bit alignment is reached. For this reason "signature" fields are replaced by "variable size signature + padding" fields and a "signature size" field is added to the three timestamp exchange messages, in order to distinguish the actual signature from the padding bits. Furthermore, digest computations involving the symmetric shared secret between nodes are replaced by PGP signatures derived from private keys on board of the network nodes, and,

[2]See chapter 4 on policy based routing.

[3]See section 3.1 on Pretty Good Privacy (PGP).

because the signature length is not known in advance, a field filling process is introduced prior to the actual signature computation.

The structure of the new OLSR messages is described in figure 5.1, figure 5.2, figure 5.3 and figure 5.4.

The timestamp exchange mechanism, as stated before, similar to the one used in [2], but adapted to asymmetric key cryptography, assumes that nodes' local clocks are *relatively synchronized*, i.e. run with the same frequency, and can be summarized as follows (refer also to figure 5.6):

1. challenge: $A \rightarrow B : Ch_a S(M)$;
2. challenge-response: $B \rightarrow A : Ch_b Ts_b D(IP_b, Ch_a) S(M)$;
3. response-response: $A \rightarrow B : Ts_a D(IP_a, Ch_b) S(M)$;

where:

- Ch_x is a 32-bit nonce generated by node X;
- $S(s)$ the PGP signature of data s computed using a local private key;
- $D(d_1, d_2, \ldots)$ the digest of the concatenation of $d_1, d2, \ldots$;
- Ts_x the timestamp of node X;
- IP_x the main address of node X;
- M the message, with the "Message Size" and the "Signature Size" fields filled using the first two bytes of the "Timestamp" field, except in challenge messages, where the first two bytes of the "Challenge" field are used instead.

Figure 5.5 shows a flow chart describing the algorithm followed on the arrival of a signed packet. Signature verification is an CPU intensive operation, so the need for it is minimized by checking in advance:

- **if a timestamp is registered for the originator**. If there is not a registered timestamp associated to the originator, then drop the

0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Scheme	Algorithms	Reserved
Timestamp		
Signature (variable size)		

Figure 5.1: Web of Trust OLSR extension basic signature message

0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Destination		
Random value "challenge"		
Signature size		
Signature (variable size) + Padding		

Figure 5.2: Web of Trust OLSR extension timestamp exchange challenge message

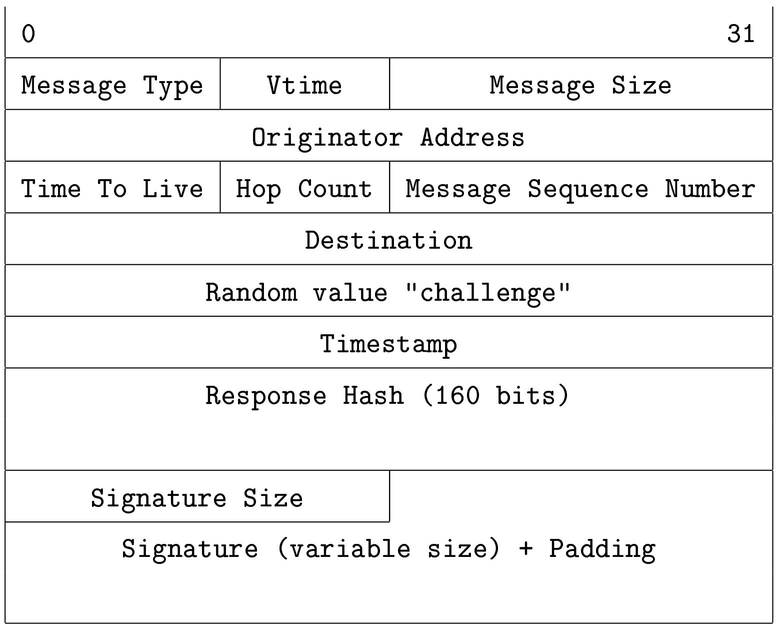


Figure 5.3: Web of Trust OLSR extension timestamp exchange challenge-response message

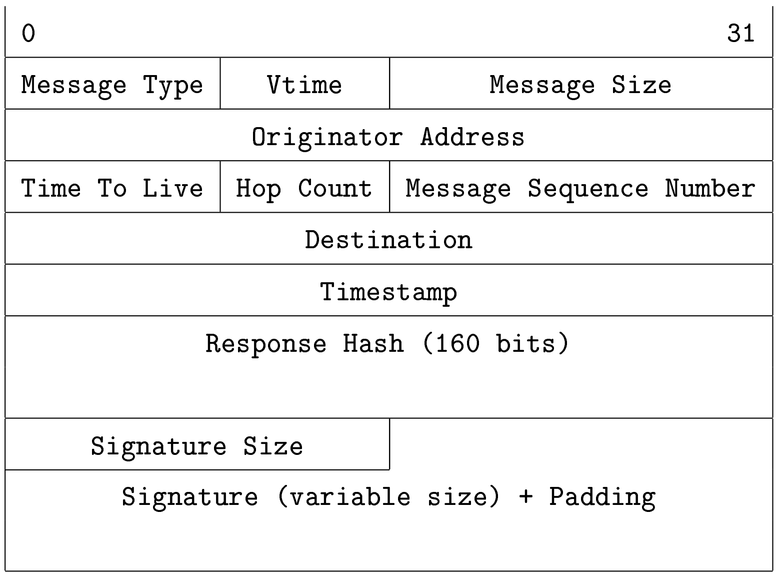


Figure 5.4: Web of Trust OLSR extension timestamp exchange response-response message

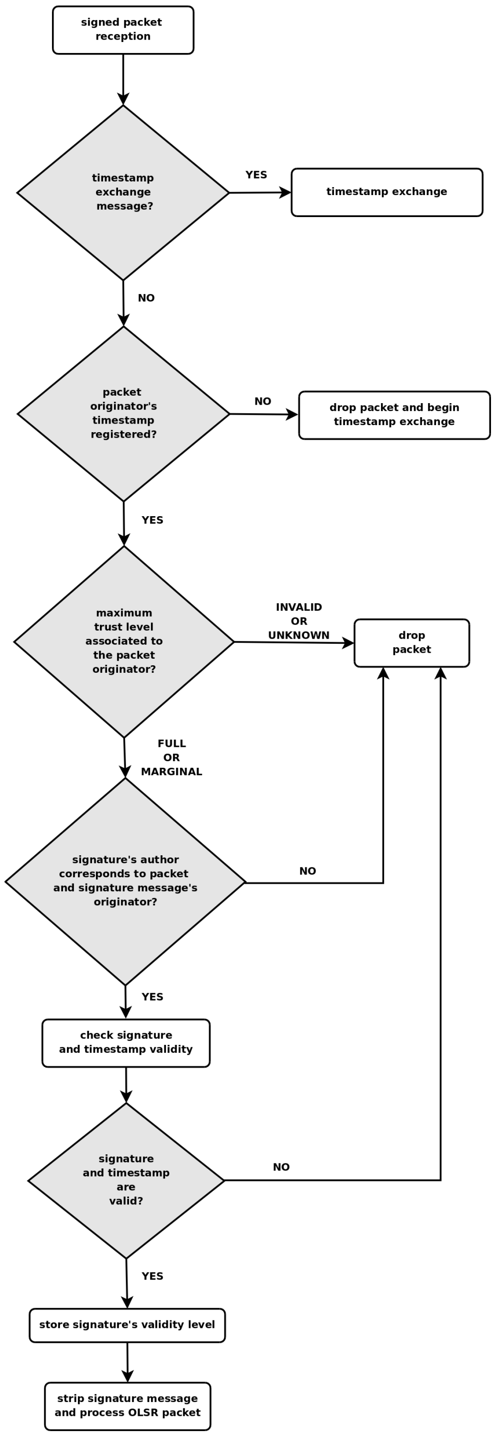

signed packet reception
timestamp exchange message?
YES
timestamp exchange
NO
packet originator's timestamp registered?
NO
drop packet and begin timestamp exchange
YES
maximum trust level associated to the packet originator?
INVALID OR UNKNOWN
drop packet
FULL OR MARGINAL
signature's author corresponds to packet and signature message's originator?
NO
YES
check signature and timestamp validity
signature and timestamp are valid?
NO
YES
store signature's validity level
strip signature message and process OLSR packet

Figure 5.5: A flow chart describing the algorithm used upon the arrival of a new signed packet.

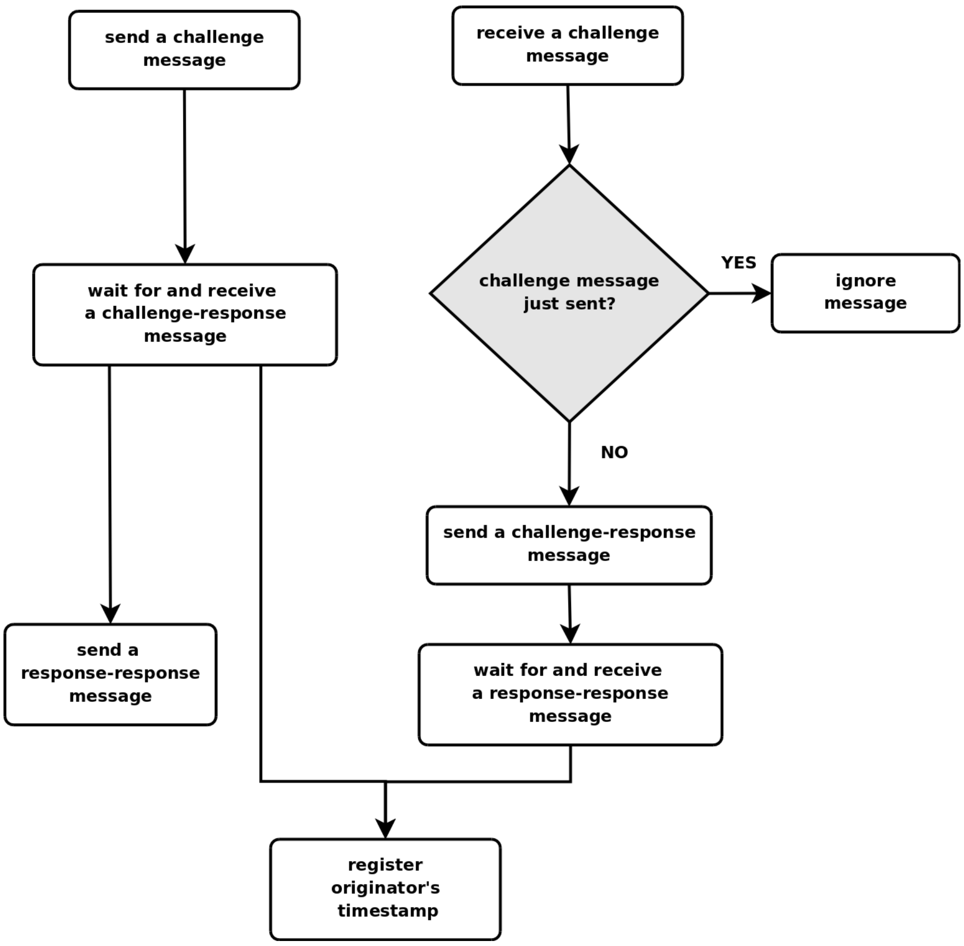

send a challenge message
wait for and receive a challenge-response message
send a response-response message
receive a challenge message
challenge message just sent?
YES
ignore message
NO
send a challenge-response message
wait for and receive a response-response message
register originator's timestamp

Figure 5.6: A flow chart of the timestamp exchange algorithm.

received packet and begin a timestamp exchange by sending a challenge message;

- **the maximum level of trust associated to the message originator**. Several PGP signatures may be associated to the same message originator, each with a trust level. If the highest level of trust associated to these signatures corresponds to "invalid" or "unknown", then the signature can be considered "invalid" without the need for cryptographic verification;
- **that the author of the signature corresponds to the message originator**.

In the basic signature message (figure 5.1), a 32-bit **Timestamp** field is used to prevent replay attacks. Message sequence numbers are not enough for this task: with 16 bits wraparound occurs too frequently[4]. A **Variable Size Signature** field follows. The signature is obtained by copying the contents of the first two bytes of the "Timestamp" field into the "Message Size" and OLSR "Packet Length" fields, and then computing, with a local private key, the PGP signature of:

- the OLSR packet header, with the modified "Packet Length" field,
- all OLSR messages in the packet except the signature message,
- the header of the signature message, with the modified "Message Size" field,
- the sub header of the signature message,
- the timestamp.

The basic signature message must be the last message in the OLSR packet, so no considerations on 32-bit alignment of the payload are needed.

The three timestamp-exchange messages are self-signed, as basic signature messages cannot be employed without consequences on security if the

[4]See section 1.3.

timestamp exchange has not been performed. In fact basic signature messages rely on the "Timestamp" field to avoid replay attacks, thus, if the timestamp exchange is not performed previously, a receiver cannot be sure of the validity of the timestamp and may be vulnerable to replay attacks.

Moreover, when an OLSR packet from a neighbor whose timestamp has not yet been registered is received, the timestamp exchange messages are extracted from the packet and processed, while the other messages, including the signature message, are dropped.

Because variable-size signatures, whose size is not known in advance, are used, some considerations are needed:

- before signing or verifying *challenge-response* (figure 5.3) or *response-response* (figure 5.4) messages, their "Message Size" and "Signature Size" fields are filled with the first two bytes of the "Timestamp" field. No "Timestamp" field is present in *challenge* messages (figure 5.2), so the first two bytes of the "Challenge" field are used for the same purpose [5];

- after signing, the "Signature Size" field is updated with the size of the signature, the end of the message is filled with padding as needed in order to align it on 32-bits, and the "Message Size" field is updated with the number of bytes measured from the beginning of the "Message Type" field till the end of the padding (or the signature, if padding is not present).

When the timestamp-exchange process is completed, basic signature messages may be employed. Upon the reception of a packet with a basic signature message, the validity of the signature is checked as well as the fact that the message originator's address corresponds to the author of the signature. If the signature is not valid or the message originator is not the

[5] "Timestamp" and "Challenge" fields are chosen as filling instead of constant values, for the purpose of adding entropy to the signature.

author of the signature, the packet is dropped. Otherwise the validity value[6] associated to the signature is stored in a "trust state" variable.

After OLSR packet processing, instead of a single standard routing table, taking advantage of policy routing features, multiple routing tables are updated. The selection of which routing table to use for each entry is based on the value of the "trust state" variable.

5.2 Security Considerations

In this section, the security considerations exposed in section 1.2 are completed with further observations, at the light of the afore-made description of the *Web of Trust OLSR extension*.

The attacks that the extension aims to counter (§ figure 1.6) are *generation of incorrect control messages* and *replay attacks*.

Generation of incorrect control messages, i.e. identity and link spoofing, is opposed by the use of a signature message in all OLSR control traffic packets. The assumption made is that the nodes in the network are not compromised and that trusted nodes are not malicious.

Replay attacks are prevented by the use of a timestamp in the signature message, and, to avoid the necessity for time synchronization between nodes, a timestamp-exchange mechanism is introduced.

In timestamp-exchange messages, the values of the "Signature Size" and "Message Size" fields are not included when computing the signature, thus they are not verified for integrity and could be tainted through a *man in the middle* attack. We here suppose that the physical integrity of the nodes and the secrecy of stored private keys have not been compromised.

The simple modification of the above-cited fields by an attacker should have the only effect of affecting the set of bytes in the packet considered as a signature by the receiver. This set of bytes would be a valid signature with respect to the originator with probability very close to zero, thus certainly

[6]The PGP validity value, as described in section 3.2, may assume three different values: "Invalid", "Marginally valid" and "Valid".

the packet would be dropped by the receiver after unsuccessful signature verification.

If in addition to the “Signature Size” and “Message Size” fields, the signature or the final padding are modified, asymmetric cryptography’s properties render the task of creating a valid new signature, without knowing the private key of the originator, practically impossible. So also in this case, the packet would be dropped by the receiver.

Having the receiver drop packets that were correctly formed by the originator would result in a *Denial of Service (DoS)*. But this may be also obtained by tainting every other byte in the message, hence the proposed solution for managing variable-size signatures does not add extra vulnerabilities with respect to the described scenario.

Chapter 6

Implementing and Testing the OLSR Web of Trust Extension

This chapter reports the implementation and subsequent testing work, done for this thesis, of the OLSR Web of Trust Extension described in chapter 5.

6.1 Implementation

The Web of Trust OLSR extension has been developed as a plugin of *olsrd*, the UniK OLSR Implementation[1], and intervenes in various spots of the OLSR packet processing algorithm. The plugin, named "Web of Trust Plugin", registers its functions using the interface provided by olsrd (§ figure 6.1 and figure 5.5).

The olsrd daemon passes the control to the Web of Trust Plugin functions upon the following events:

- **when a packet is received**, to check if a timestamp is registered with the packet's originator. If a timestamp is not registered, then the

[1]See section 1.4.

timestamp-exchange process is initiated. Otherwise, if a tmestamp is already registered:

  - the maximum level of trust associated with the originator is retrieved from the local GPG database. If the maximum trust level corresponds to "invalid", i.e. the stored public key associated to the originator is outdated or untrusted, then the entire packet is discarded;
  - the packet's signature, contained in the signature message, is checked for correctness. If the signature is incorrect, then the packet is discarded;
  - from the originator's IP address, the corresponding e-mail address is retrieved[2]. This is checked against the author of the signature. If it does not match, then the packet is discarded;
  - if the packet passed all the tests, then the trust (validity) value associated to the originator is stored in a "trust state" variable. Then control passes again to olsrd.

- **when an entry is added to the routing table**, the plugin populates three different routing tables, basing itself on the "trust state" value determined in the previous steps. Control passes back to olsrd;
- **before an OLSR packet is sent**, a signature message is affixed, using the algorithm afore-described in section 5.1, and control passes again to olsrd.

The Web of Trust Plugin is written in C with some external dependencies:

- The GnuPG Made Easy (GPGME) library (briefly introduced in section 3.3.1), which in turn depends on:

[2]The association between IP addresses and e-mail addresses should be supplied by the user through the olsrd configuration file.

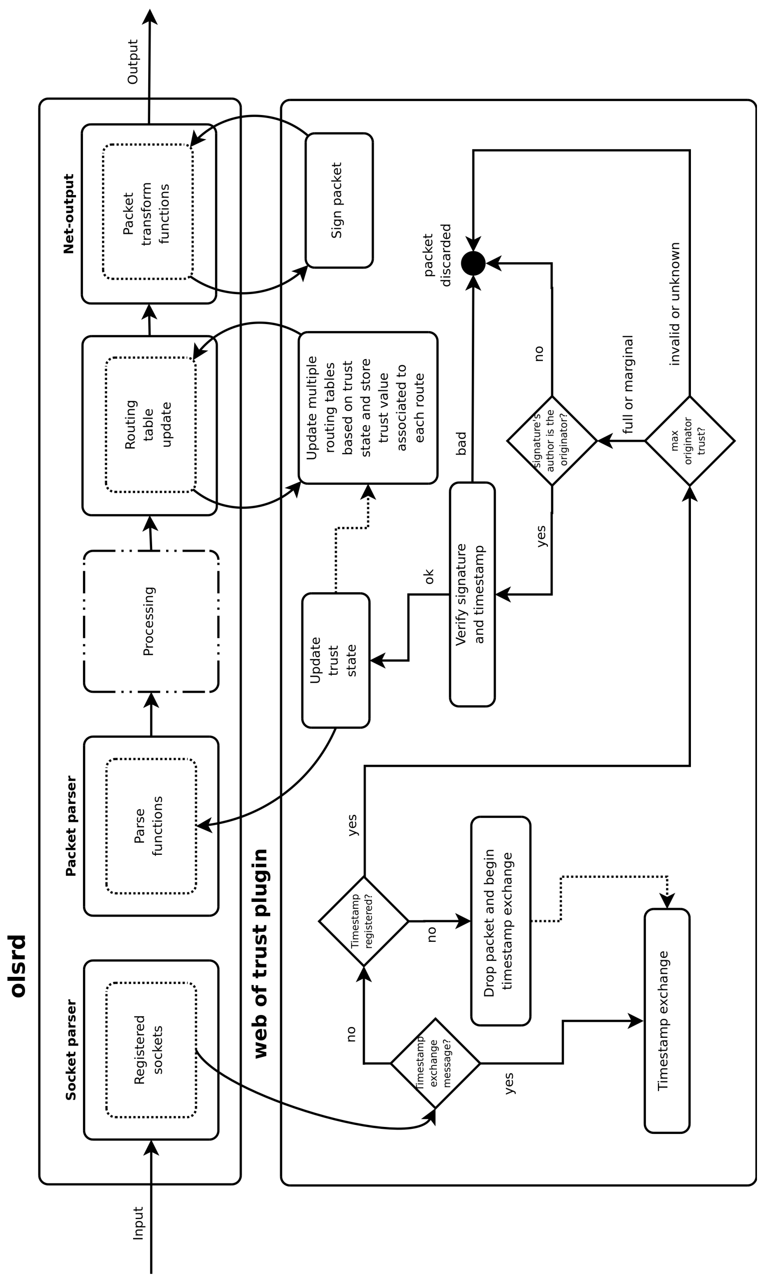


Figure 6.1: The OLSR Web of Trust plugin design.

– GNU Privacy Guard (GnuPG) (see chapter 3);

– the libgpg-error library, that can be freely downloaded from the GnuPG website [40].

As Linux routing policy database (RPDB) calls are employed by the implementation, the plugin is compatible only with GNU/Linux systems.

6.2 Configuration

In order to enable the Web of Trust plugin, a new section has to be added to the olsrd configuration file, and some parameters should be specified. A sample configuration section is here reported, along with an explanation of the plugin configuration parameters.

```
LoadPlugin "olsrd_wot.so.0.1"
{
      PlParam "Keyname" "node@ninux.org"
      PlParam "Gpghomedir" "/root/.gnupg"
      PlParam "Gpgfilename" "/usr/bin/gpg"
      PlParam "Passphrase" "node"
      #PlParam Passphrasehelper" "/root/passhelper.sh"

      PlParam "Ipowner" "172.20.0.9 source@ninux.org"
      PlParam "Ipowner" "172.20.0.17 source@ninux.org"
      PlParam "Ipowner" "172.20.0.18 node2@ninux.org"
      PlParam "Ipowner" "172.20.0.25 node2@ninux.org"
      PlParam "Ipowner" "172.20.0.1 node2@ninux.org"
}
```

The configuration parameters that may be specified are:

- **Keyname**: the e-mail, name or key id associated to the PGP private key used to sign the packets;
- **Gpghomedir**: the GnuPG home directory;
- **Gpgfilename**: the full path of the GnuPG binary;

- **Passphrase**: the pass phrase needed to use the private key;
- **Passphrasehelper**: if the user is not willing to provide a cleartext pass phrase, she may provide the full path of a program that outputs the pass phrase;
- **Ipowner**: specify an IP to e-mail association in order to retrieve the appropriate public key associated to the sender of a packet. At the moment only one e-mail can be associated to an IP address.

Furthermore, in order to permit a correct operation of the extended protocol, some OLSR parameters had to be empirically adapted. The new recommended values, to be specified in the olsrd configuration file, are summarized in table 6.1.

Parameter name	RFC 3236 value	new value
HelloInterval	2.0	6.0
HelloValidityTime	6.0	60.0
TcInterval	5.0	15.0
TcValidityTime	15.0	75.0
MidInterval	from 5.0	15.0
MidValidityTime	15.0	75.0
HnaInterval	5.0	15.0
HnaValidityTime	15.0	75.0

Table 6.1: New recommended values for some OLSR parameters.

In order to take advantage of the policy routing features of Linux, some rules have to be added to the RPDB (§ chapter 4), by using the *iproute* [46] tool. The routing tables currently used (but may change, or better be configurable by the user in the future) by the Web of Trust plugin are table 210 for fully trusted entries and table 220 for marginally **and** fully trusted entries, and table 254 (table main) that contains all routes.

Using the following rules, the most trusted route available will be selected for incoming traffic:

```
ip rule add from all priority 30010 table 210
ip rule add from all priority 30020 table 220
```

Or a minimum granted service semantic may be defined on the "Type of Service" field of IP packets:

```
ip rule add tos 0x0f priority 30010 table 210
ip rule add tos 0x0f priority 30015 blackhole  #if no match is
     found in table 210, silently drop the packet
ip rule add tos 0x1f priority 30020 table 220
ip rule add tos 0x1f priority 30025 blackhole  #if no match is
     found in table 220, silently drop the packet
```

6.3 Testing

During development, tests were performed using the Netkit [48, 49] network emulator. This section will show how to set up a testing environment for the Web of Trust olsrd plugin and the obtained results.

Netkit and GnuPG[3] are assumed to be correctly installed on the system, which is supposed to be GNU/Linux.

Two virtual machines directly connected, `node1` and `node2`, are set up, through the following steps:

1. Install GNU Privacy Guard and the GnuPG Made Easy library in the virtual machines' directories. The compressed source code of the two packages is supposed to be already downloaded, their checksums checked, and stored in the user's home (˜) directory:

```
$ mkdir -p wotlab/shared/usr/local
$ cd wotlab/
$ WOT_PREFIXDIR="`pwd`"

$ tar -jxf ~/gnupg-1.4.9.tar.bz2
```

[3] For more information on Netkit and GNU Privacy Guard (GPG) please refer to [49], section 3.3 and [40].

```
$ cd gnupg-1.4.9/
$ ./configure --prefix=$WOT_PREFIXDIR/shared/usr/local
$ make
$ make install
$ cd ..
$ rm -rf gnupg-1.4.9/

$ tar -jxf ~/gpgme-1.1.4.tar.bz2
$ cd gpgme-1.1.4/
$ ./configure --prefix=$WOT_PREFIXDIR/shared/usr/local
$ make
$ make install
$ cd ..
$ rm -rf gpgme-1.1.4/
```

2. Install Olsrd, patched with the wot plugin patch, available from [50] too[4]:

```
$ wget http://stud.netgroup.uniroma2.it/~claudio.pisa
                /wotplugin/addwot-olsrd-0.5.6-r2.patch
$ tar -jxf ~/olsrd-0.5.6-r2.tar.bz2
$ cd olsrd-0.5.6-r2/
$ patch -p1 < ../addwot-olsrd-0.5.6-r2.patch
$ make OS=linux build_all
$ make DESTDIR=$WOT_PREFIXDIR/shared install_all
$ cd ..
$ rm -rf olsrd-0.5.6-r2
$ rm addwot-olsrd-0.5.6-r2.patch
```

3. Generate the PGP keys and create a minimal web of trust:

```
$ mkdir -p node1/root/.gnupg
$ mkdir -p node2/root/.gnupg

$ gpg --homedir node1/root/.gnupg --gen-key
```

[4]The author hopes that the Web of Trust Plugin source code will be merged in some future release of olsrd, making the patching step useless.

```
# Choose DSA and Elgamal, 1024 bits key size, no
    expiration, real name "Foo Foo", e-mail address
    node1@ninux.org, pass phrase "foo"

$ gpg --homedir node2/root/.gnupg --gen-key
# Choose DSA and Elgamal, 1024 bits key size, no
    expiration, real name "Bar Bar", e-mail address
    node2@ninux.org, pass phrase "bar"

$ gpg --homedir node1/root/.gnupg --export --output
    node1export
$ gpg --homedir node2/root/.gnupg --export --output
    node2export
$ gpg --homedir node1/root/.gnupg --import node2export
$ gpg --homedir node2/root/.gnupg --import node1export
$ rm node?export

# Perform normal signatures, fully trusting the other user
$ gpg --homedir node1/root/.gnupg --sign-key "Bar Bar"
$ gpg --homedir node2/root/.gnupg --sign-key "Foo Foo"

# To assign a trust value to a key owner, use the
    following syntax instead:
# $ gpg --homedir node1/root/.gnupg --edit-key "Bar Bar"
# and choose tsign
```

4. Create a new file called `lab.conf` and add the following:

```
LAB_AUTHOR="Claudio Pisa"
LAB_EMAIL="clauz A T ninux.org"
LAB_DESCRIPTION="Web of Trust OLSR extension test"

node1[0]=A
node2[0]=A
```

5. The same for `node1.startup`:

```
/sbin/ip addr add 10.0.0.1/16 dev eth0 broadcast
    10.0.255.255
```

```
/sbin/ip link set eth0 up
/sbin/ldconfig
```

6. and `node2.startup`:

```
/sbin/ip addr add 10.0.0.2/16 dev eth0 broadcast
    10.0.255.255
/sbin/ip link set eth0 up
/sbin/ldconfig
```

7. Create `etc` directories and copy the `olsrd.conf` file:

```
$ mkdir -p node1/etc
$ mkdir -p node2/etc
$ cp shared/etc/olsrd.conf node1/etc
$ cp shared/etc/olsrd.conf node2/etc
```

8. Edit `node1/etc/olsrd.conf`, adding the following lines:

```
LoadPlugin "olsrd_wot.so.0.1" {
        PlParam "Keyname" "Foo"
        PlParam "passphrase" "foo"
        PlParam "gpghomedir" "/root/.gnupg"
        PlParam "gpgfilename" "/usr/bin/gpg"
        PlParam "Ipowner" "10.0.0.2 node2@ninux.org"
}
```

   Specify the interface `"eth0"` instead of `"XXX" "YYY"`, and change the olsr parameters as specified in table 6.1.

9. And also edit `node2/etc/olsrd.conf`:

```
LoadPlugin "olsrd_wot.so.0.1" {
        PlParam "Keyname" "Bar"
        PlParam "passphrase" "bar"
        PlParam "gpghomedir" "/root/.gnupg"
        PlParam "gpgfilename" "/usr/bin/gpg"
        PlParam "Ipowner" "10.0.0.1 node1@ninux.org"
}
```

Specifying here too the `eth0` interface and changing the OLSR parameters.

10. Then launch the Netkit lab:

```
$ lstart
```

11. Run the olsrd daemon on both virtual machines

```
$ olsrd
```

In table 6.2 some measures from an emulation performed using Netkit running on a computer with an AMD64 2GHz CPU are provided. The "Symmetric" column refers to the Secure Extension to the OLSR Protocol described in [2], which uses 128 bit long keys, which is a good trade-off between performance and security using symmetric cryptography. In order to provide a similar level of security with asymmetric cryptography, to the author's advice at least 1024 bits long keys should be employed.

	Symmetric	RSA	DSA	measure unit
Length of the key	128	1024	1024	bits
Basic signature verification	0.132	159.422	150.784	milliseconds
Basic signature adding	0.05	55.477	60.130	milliseconds
Challenge verification	0.689	93.274	98.664	milliseconds
Challenge-response verification	0.759	170.079	196.656	milliseconds
Challenge-response creation	0.522	45.188	49.327	milliseconds
response-response verification	0.141	175.373	194.526	milliseconds
response-response creation	0.532	78.992	82.78	milliseconds

Table 6.2: Measured times for some cryptographic operations

Chapter 7

Conclusions, Current and Future Work

7.1 Conclusions

In this thesis, the **Web of Trust OLSR Extension**, a security extension to the OLSR protocol for multi-administered networks, was introduced.

Multi-administered networks, especially wireless community networks, are spreading and have already become critical infrastructures in several areas of the world. Many of them employ the Optimized Link State Routing (OLSR) protocol for Mobile Ad Hoc Networks (MANETs) to manage their routing infrastructure. As these networks develop, they may become target of various threats, and thus the need for security at the infrastructure level emerges. Some of the existing OLSR security extensions rely on symmetric cryptography and a shared secret between the nodes of the network, but this is not acceptable in community networks, where an administrative domain may be as small as one node, and there is very loose coordination between network administrators. In this scenario, asymmetric cryptography and reputation systems have to come into play. A widely accepted standard in both of these areas is represented by PGP and its web of trust model.

By using hop-by-hop OpenPGP signatures in OLSR control packets, and an appropriate timestamp exchange mechanism, some threats, such as identity spoofing, link spoofing and replay attacks, may be countered by the Web of trust extension. No protection is provided for user traffic, which may be secured by other means (for example by using TLS/SSL).

Moreover, by employing policy based routing, and updating multiple routing tables on packet signature validity basis, paths corresponding to different levels of trust may be traced over the routing infrastructure.

The Web of Trust extension has been implemented as a plugin of the UniK OLSR Implementation (olsrd), popular among community network members, and subsequently tested in an emulated environment.

7.2 Current Work

Departing from the work done for this thesis, an article entitled "Trusted Routing in Wireless Community Mesh Networks" has been derived. This article is reported in appendix A, at the time of this writing is in the process of being completed for the IEEE International Conference on Communications 2009 [51], and a similar article is planned to be submitted for publication to the Security and Communication Networks Journal (Wiley) [52] in the next months.

7.3 Future Work

The well-known slowness of asymmetric cryptography with respect to symmetric cryptography is reaffirmed by the performed tests. This slowness, produced by CPU time consumption and which should be minimized especially for an employment of the plugin on board of embedded devices, could be mitigated by the use of a symmetric key exchange mechanism between neighbors, in addition to the timestamp exchange. This mechanism could be designed as follows: when node A has performed a timestamp exchange with node B, it generates a random session key. This session key is then encrypted with the public key of the administrator of node B, and sent to node

B. Node B generates a random session key too, and sends it, encrypted with the public key of the administrator of node A, to node A. The key exchange is done after the timestamp exchange in order to prevent replay attacks on the key exchange protocol. Each node then uses the session key received by its neighbor X to sign, using *symmetric* cryptography (e.g. SHA-256 hashes, but the specific algorithm could be – better – negotiated), OLSR packets addressed to X, and the session key itself has choose to verify the signatures affixed to the OLSR packets received from its neighbor. It should be pointed out that in the depicted mechanism, a node generates and stores a session key for each trusted neighbor, each key associated to an incoming control traffic stream.

Moreover, to save more CPU time, perhaps the cryptographic digests used in the timestamp exchange process could be, after an accurate study, removed and replaced with the concatenation of their cleartext arguments.

PGP key distribution currently relies on the nodes' administrators, which should share their keys among themselves, assign reputation levels through key signing, and update their node's trust base. The use of PGP keyservers, located in various spots of the network, containing, as usual for PGP keyservers, the same, replicated information, could be coupled with an automatic key retrieving and updating mechanism from the network nodes. The major drawback is that in case of a network split[1], synchronization between the keyservers could be lost until the network is re-united. A system based on DHT (Distributed Hash Tables) coupled with an appropriate protocol, could be devised in order to obtain a highly decentralized key distribution mechanism without the need for keyservers.

To take better advantage of the trusted paths located by the Web of Trust extension, a well-determined semantic could be defined on the "Type of Service" field of the IP packets belonging to user traffic. Some values could then be associated to a trust maximization service, while others to a

[1]Network splits may occur frequently in wireless community networks, due to the unreliability of the wireless medium.

speed maximization service.

Furthermore, an user-friendly system could be designed, with the capability of translating Pico Peering Agreement (or similar agreements) clauses into policy routing rules.

If address autoconfiguration mechanisms are employed, no centralized IP address allocation is needed, and thus users may not know the association between IP addresses and network node's administrators. In this case, the step in which the author of the signature is checked against the message originator can be simply skipped.

Extensions similar to the Web of Trust Extension could be devised for other protocols that operate in multi-administered networks; for example B.A.T.M.A.N., AODV, or even BGP.

The plugin, that relies on the GnuPG Made Easy (GPGME) library for signature and key manipulation, works with several versions of olsrd, but the use of Linux RPDB (Routing Policy Data Base) specific calls restricts the portability of the implementation to GNU/Linux systems. Other operating systems, such as FreeBSD, support policy routing in their kernels, and thus the plugin might be ported to these systems in a future work. On those platforms where policy routing support is not available but GPGME is operational, the policy routing features might be simply deactivated in the plugin, in order to at least take advantage of the protection provided by signatures affixed to control packets.

Currently, in the implementation, routes output by the standard OLSR packet processing algorithm are added to the routing table determined by the validity value computed in the most recent signature check. Thus trust is not used as a metric, nor affects the MPR selection process, but these features could be implemented in the future.

IPv6 support is not implemented yet, but, as olsrd fully supports IPv6, it could be easily added with minor changes to the codebase.

The plugin has only been ran in an emulated environment. Tests in a real environment, especially in a community network, would be very interesting.

Appendix A

Trusted Routing in Wireless Community Mesh Networks

In this appendix, an article entitled "Trusted Routing in Wireless Community Mesh Networks", derived from the work done for this thesis, is reported. At the time of this writing is in the process of being completed for the IEEE International Conference on Communications 2009 [51], and a similar article is planned to be submitted for publication to the Security and Communication Networks Journal (Wiley) [52] in the next months.

Trusted Routing in Wireless Community Mesh Networks

Claudio Pisa
Wireless community network member
Ninux.org
Rome, Italy
Email: clauz@ninux.org

Francesco Saverio Proto
Ph.D. Student
Università degli studi di Roma Tor Vergata
Rome, Italy
Email: proto@ing.uniroma2.it

***Abstract*—The routing control traffic in a wireless community mesh network crosses administrative boundaries. The network is partitioned in many administrative domains, sometimes as little as one node, connected by autoconfigured radio links. Because neighboring domains are typically unknown, have an authentic and authorized routing information is an open problem. In this paper we propose a novel approach to authenticate routing control traffic in wireless community networks, taking into account trust and reputation between network nodes in different administrative domains.**

INTRODUCTION

Routing protocols are target of many attacks [1]. Appropriate security extensions are necessary, to make sure information in the routers' databases is authentic and authorized. A routing domain managed by an IGP protocol usually corresponds to a single administrative domain. For this reason IGP routing protocols security extensions are based on a pre-shared secret between the nodes. This creates an open problem because it is not possible to deploy pre-shared keys in wireless community networks where every node is owned by a different entity.

Reputation of a remote (possibly unknown) node to decide to accept its advertised routing information is necessary. Alternatives to a pre-shared secret is a PKI model based on centralized third parties as the one proposed in [2], but this is not desirable in multi-owned network infrastuctures without a central administration. In wireless mesh networks it is not trivial to deploy a centralized security mechanism because network splits are frequent due to the unreliability of the wireless medium.

We propose a novel approach, *trusted routing*, where unknown nodes can exchange route information with a certain level of confidence. We make use of a reputation framework: the network administrators create a web of trust among themselves without the need for central entities.

We enhance network nodes with the capability of deciding whether to trust or not routing information received from other nodes. The routing information is not just accepted or rejected: according to a metric, the information collected is classified with a trust level and handled according to local policies.

In our work we extend the OLSR routing protocol, widely used by European wireless community networks, with our Web of Trust security extension.

In the remainder of the paper we show our proposed extension to the OLSR routing protocol, going through a security analisys. At last we show the details of our implementation and testing and we draw our conclusions.

I. PROBLEM STATEMENT

Wireless community networks are intrinsically decentralized with only minimal coordination. The goal of these networks is to make possible to not technical people (community members) to deploy their own infrastructure.

Community members deploy antennas to exchange data with other unknown members in radio proximity. Consequently the organization of the network happens by distributed autoconfiguration. Radio links are automatically exploited when neighbours nodes are detected. Recent research [3], [4], [5] at the network layer identified solutions to automatically configure a network without the need of any prior coordination.

The lack of a central authority managing the network as a whole makes impossible the adoption of security extensions based on a pre-shared secret installed on all the nodes. If routing protocols for wireless mesh networks are not extended with the capability of assigning a reputation to unknown nodes, routing control traffic cannot be authenticated and authorized, and thus cannot be trusted.

To the best of our knowledge there is not any decentralized security extension to routing protocols, capable of managing the trust relationships between unknown network nodes.

In this work we present a security extension to the OLSR routing protocol to manage reputation of unknown nodes directly at the network level, without the involvement of central authorities. Our work is based on the well consolided standard PGP reputation framework.

II. PROTOCOL

We extended the OLSR routing protocol with a security extension based on a Web of Trust.

A. Overview

We briefly highlight the characterics of the OLSR protocol functional to our discussion, omitting a complete overview that can be found in [6].

OLSR packets (figure 1) are composed of one or more OLSR messages. It is possibile to define new OLSR messages to add new features to the base protocol.

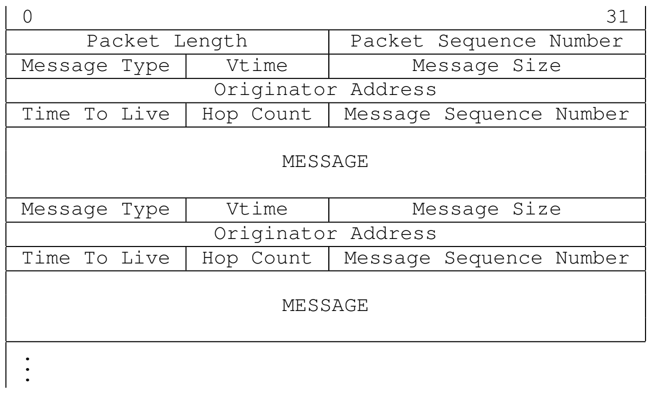


Fig. 1. Basic OLSR Packet Format

To create the topology of the network, OLSR sends over the media link broadcast packets that any other node in radio proximity can receive. Two are the foundamental messages of the OLSR protocol: HELLO, and TC. HELLO messages are used for neighbor discovery and link sensing; these packets expire after one hop and are never forwarded. TC messages are used for network topology information diffusion; these packets are forwarded away from the originator to deliver topology information incapsulated into new OLSR packets at each hop.

A node running our Web of Trust OLSR extension is associated to a public/private key pair of his administrator. The node signs with this private key all the generated OLSR packets. Packets not signed, or with an invalid signature, are immediately discarded by the receivers. The information received in OLSR packets with a correctly verified signature is evaluated according to local trust policies. The Web of Trust extension is responsible for three tasks: i) signature and verification of OLSR packets, ii) synchronization of the clocks between neighbor nodes to prevent replay attacks, iii) and the enforcement of local trust policies on the collected routing information. The process of sending and collect routing information is here summarized:

- **when a packet is received**, the node checks if a timestamp is registered with the packet's originator. If a timestamp is not registered, then the timestamp-exchange process is initiated to find the time difference between the local clock and the new neighbor's clock. Otherwise:
  - the maximum level of trust associated with the originator is retrieved from the local PGP database. If the maximum trust level corresponds to "invalid", i.e. the stored public key associated to the originator is unknown or outdated or untrusted, then the entire packet is discarded;
  - the packet's signature, contained in the signature message, is checked for correctness. If the signature is incorrect, then the packet is discarded;
  - if the packet passed all the tests, then the trust (validity) value associated to the originator is stored in a "trust state" variable.
- **before an OLSR packet is sent**, a signature message is affixed, containing a signature of the all packet. The signature is obtained by copying the contents of the first two bytes of the Timestamp field into the "Message Size" and "OLSR Packet Length" fields, and then computing, with a local private key, the PGP signature of:
  - the OLSR packet header, with the modified "Packet Length" field,
  - all OLSR messages in the packet except the signature message,
  - the header of the signature message, with the modified "Message Size" field,
  - the sub header of the signature message,
  - the timestamp.

B. The Timestamp Exchange

To prevent replay attacks we introduce in the OLSR signature message a 32 bit timestamp. We do not use the existing OLSR sequence number because a 16 bit counter does not protect from replay attacks, as explained in detail in [7].

The signaling employed to register the nodes' timestamps assumes that nodes' local clocks are *relatively synchronized*, i.e. run with the same frequency, and can be summarized as follows:

1) challenge: $A \to B : Ch_a S(M)$;
2) challenge-response: $B \to A : Ch_b Ts_b D(IP_b, Ch_a) S(M)$;
3) response-response: $A \to B : Ts_a D(IP_a, Ch_b) S(M)$;

where:

- Ch_x is a 32-bit nonce generated by node X;
- $S(s)$ the PGP signature of data s computed using a local private key;
- $D(d_1, d_2, \ldots)$ the digest of the concatenation of $d_1, d2, \ldots$;
- Ts_x the timestamp of node X;
- IP_x the main address of node X;
- M the message, with the "Message Size" and the "Signature Size" fields filled with the first two bytes of the "Timestamp" field, except for challenge messages, in which the first two bytes of the "Challenge" field are used for the same purpose.

Note that when a node receives an OLSR packet with any timestamp-exchange message, the rest of the messages in the packet are discarded. In fact timestamp-messages are self-signed while any other message in the received packet does not have a valid signature. To cope with variable size signatures we used the following rules:

- before signing or verifying *challenge-response* (figure 3) or *response-response* (figure 4) messages, their "Message Size" and "Signature Size" fields are filled with the first two bytes of the "Timestamp" field. No "Timestamp" field is present in *challenge* messages (figure 2), so the

0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Destination		
Random value "challenge"		
Signature size		
Signature (variable size) + Padding		

Fig. 2. Web of Trust OLSR extension timestamp exchange challenge message

0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Destination		
Random value "challenge"		
Timestamp		
Response Hash (160 bits)		
Signature Size		
Signature (variable size) + Padding		

Fig. 3. Web of Trust OLSR extension timestamp exchange challenge-response message

0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Destination		
Timestamp		
Response Hash (160 bits)		
Signature Size		
Signature (variable size) + Padding		

Fig. 4. Web of Trust OLSR extension timestamp exchange response-response message

first two bytes of the "Challenge" field are used for the same purpose [1];
- after signing, the "Signature Size" field is updated with the size of the signature, the end of the message is filled with padding as needed in order to align it on 32-bits, and the "Message Size" field is updated with the number of bytes measured from the beginning of the "Message Type" field till the end of the padding (or the signature, if padding is not present).

C. New message types

To extend the protocol with the new functionalities we had to define four new message types.

- Challenge Message
- Challenge Reponse Message
- Response Response Message
- Basic Signature Message

[1]"Timestamp" and "Challenge" fields are chosen as filling instead of constant values, for the purpose of adding entropy to the signature.

0		31
Message Type	Vtime	Message Size
Originator Address		
Time To Live	Hop Count	Message Sequence Number
Scheme	Algorithms	Reserved
Timestamp		
Signature (variable size)		

Fig. 5. Web of Trust OLSR extension basic signature message

D. Protocol tuning

In order to permit a correct operation of the extended protocol, some OLSR parameters had to be adapted. The new recommended values are:

- HelloInterval, from 2.0 to 6.0
- HelloValidityTime, from 6.0 to 60.0
- TcInterval, from 5.0 to 15.0
- TcValidityTime, from 15.0 to 75.0
- MidInterval from 5.0 to 15.0
- MidValidityTime, from 15.0 to 75.0
- HnaInterval, from 5.0 to 15.0
- HnaValidityTime, from 15.0 to 75.0

E. Previous work

The existing security extension for OLSR [7] provides integrity for OLSR packets through digital signatures obtained using a symmetric key shared among the nodes. No confidentiality or integrity for user traffic is provided. The signature is performed at the OLSR packet level, while the OLSR messages are delivered without additional processing. OLSR packets are signed by every forwarding hop, thus the assumption made is that each forwarder trusts the previous forwarder and the source of the message. This assumption is straightforward if we consider that in [7] all nodes also share a common secret. Our Web of Trust extension provider all the features of the existing security extension for OLSR [7] but it is not necessary to rely on a pre-shared secret between all the nodes.

III. Trust model

The key idea of this paper is to use of a Web of Trust between the network nodes, to authenticate and authorize routing information without a pre-shared secret. The Web of Trust is a distributed, non-hierarchical trust model to validate public key's ownership.

Traditional, hierarchical Public Key Infrastructures (PKIs) rely on a root Certification Authority (CA), trusted by all users. The CA, by using signed electronic documents called Digital Certificates, assures the validity of public keys. A CA may act as an introducer, i.e. validate public keys directly, or as a meta-introducer, i.e. empower lower level Certification Authorities to validate keys in its place. CAs are also responsible for issuing Certificate Revocation Lists (CRLs), which include certificates whose validity has been revoked. The whole structure relies on all users' trust in the root CA and on the secrecy of the root CA's private key.

We believe that in a wireless community network without a central administration, a PKI model based on centralized third parties is not suitable.

By contrast, we based our work on the Web of Trust model [8], where each user validates other users' public keys by signing them, and thus becoming an introducer of these. When key signature takes place, a validity level and a trust level are assigned to the signed key. These values are subjective, i.e. different for every user in the web of trust.

The validity level is an indicator of how much the signing user considers the signed key to be valid. Three levels are possible:

- Full
- Marginal
- Invalid

The trust level specifies the trust the signing user puts on the signed key owner's ability in key certification. Four trust values are allowed.

- Ultimate
- Complete
- Marginal
- No trust

This makes possibile to verify a signature even if the signer is not directly known. In the Web of Trust a key is valid if a completely trusted signature or two marginally trusted signatures are affixed to it. Thus users trust other users to certify key validity, without the need for a central CA, creating a Web of Trust among themselves: a decentralized and fault-tolerant reputation system.

The administrators of the network nodes of a wireless community network create a Web of Trust among themselves[2] by signing and distributing each other's public keys. Each administrator computes a subjective trust value for every other node, based on trust rankings given by others or by herself.

In our work we apply the signature to OLSR packets. The goal of our approach is to avoid the possibility of an external maliciuos attacker of injecting forged OLSR packets into the network. Once a packet is successfully verified, its OLSR messages are considered trusted and forwarded into the mesh network. Therefore an OLSR message inside a trusted OLSR packet is considered trusted as well. Trust is not at the single message level: each node relies only on its 1-hop neighbors, and on their reputation.

Another security extension that signs OLSR messages instead of packets has been proposed in [2]. However signing every single message has different drawbacks. The most evident is on performances: in fact the number of signature and verifications each node should perform is higher. In a stable network, an OLSR packet carries, on average, 2 to 5 OLSR messages.

Moreover, signing the messages instead of the packets has also an impact on the use of the Web of Trust model.

[2]note that usually web of trust relationships between network administrators is already present, for example for email signing.

When signatures are on a per-packet basis, a node needs only to know the public keys of its 1-hop neighbours. Signing every single OLSR message means that to verify received messages, a node must store the public keys of all the nodes of the mesh network. This does not scale well with the size of the network. It may happen to have nodes discarding messages from far nodes that are not in the web of trust. This could lead to severe routing problems, as different nodes may have too different views of the network topology.

IV. Security Analisys

In this section we present a security analisys of our work following the guidelines provided in RFC 4593 [1], where generic threats to routing protocols are summarized.

With respect to the source of the threats our work focuses on outsider attackers, therefore the case of Byzantine threat sources is not in the goals of this paper.

The OLSR Web of Trust extension is able to guarantee that routing information in the routers' database is authentic and authorized. An external attacker is not able to insert bogus data in the routing database of a network node, unless she owns a trusted key.

With respect to [1] a mesh network running OLSR with the Web of Trust extension is not vulnerable to the following attacks:

- Network congestion
- Blackhole
- Starvation
- Looping
- Cut
- Delay
- Partition
- Churn
- Instability

These attacks are targeted to the network as a whole, but because it is not possibile to inject bogus data in the routers' database the attacks are not feasible. This is true as long as the private key of a network node is not compromised.

The underlaying trasport layer (UDP) is not protected by our security extension, an malicious network node may run one of the following attacks against a single node:

- Clog: a network node may starve for some resources, especially for CPU consumption, if it receives more routing protocol packets than the amount it can process. The CPU bottleneck a node may experience is due the use of public key cryptography, that requires plenty of CPU time.
- Eavesdrop: an attacker can learn the topology of the network sniffing the routing traffic, because data is signed but not encrypted.
- Overcontrol

The authors of the OLSR protocol security extension described in [7] present a security analisys covering three attacks: i) Node isolation, ii) Node spoofing and iii) Link spoofing. Regarding these attacks we can make the following considerations regarding the OLSR Web of Trust extension:

- Node isolation: this attack is possible because we have no control on the trasport layer. An attacker may also create interference on the physical link to perform node isolation.
- Node spoofing: this attack is not feasible unless the attacker has access to the private key of the node she wants to spoof.
- Link spoofing: this attack is not feasible unless the attacker has access to the private key of a trusted node.

We remark that spoofing attacks are not possibile, but node isolation is possible running a DOS attack agaist a single node.

In timestamp-exchange messages, values of "Signature Size" and "Message Size" fields are not included when computing the signature, thus they are not verified for integrity and could be tainted through a *man in the middle* attack. We here suppose that the physical integrity of the nodes and the secrecy of stored private keys have not been compromised.

The simple modification of the above-cited fields should have the only effect of affecting the set of bytes in the packet considered as a signature by the receiver. This set of bytes would be a valid signature with respect to the originator with probability very close to zero, thus certainly the packet would be dropped by the receiver after unsuccessful signature verification.

If in addition to the "Signature Size" and "Message Size" fields, the signature or the final padding are modified, asymmetric cryptography's properties render the task of creating a valid new signature, without knowing the private key of the originator, practically impossible. So also in this case, the packet would be dropped by the receiver.

Having the receiver drop packets that were correctly formed by the originator would result in a *Denial of Service (DoS)*. But this may be also obtained by tainting every other byte in the message, hence the proposed solution for managing variable-size signatures does not add extra vulnerabilities with respect to the described scenario.

V. Implementation

The implementation of the Web of Trust OLSR extension is a plugin for the UniK OLSR Implementation, also known as olsrd [9], which provides the following advantages over other implementations:

- **modularity** - new plugins can be added without changing the main codebase;
- **license** - the source code is released under a BSD-style license, i.e. it is freely available and modifiable;
- **popularity** - it is used by many *community networks* who also actively contribute to test, extend and enhance it;
- **portability** - written in C, runs on a wide range of hardware platforms (i386, ARM, MIPSEL, ...), including embedded devices, and operating systems (GNU/Linux, Mac OS, OpenWRT, *BSD, Windows, . . .).

Furthermore we use the GnuPG Made Easy (GPGME) library, part of the GNU Privacy Guard (GnuPG) [10], i.e. the GNU project's implementation of the OpenPGP standard,

TABLE I
Measured times for some cryptographic operations

	Symmetric	RSA	DSA	measure unit
Length of the key	128	1024	1024	bits
Basic signature verification	0.132	159.422	150.784	milliseconds
Basic signature adding	0.05	55.477	60.130	milliseconds
Challenge verification	0.689	93.274	98.664	milliseconds
Challenge-response verification	0.759	170.079	196.656	milliseconds
Challenge-response creation	0.522	45.188	49.327	milliseconds
response-response verification	0.141	175.373	194.526	milliseconds
response-response creation	0.532	78.992	82.78	milliseconds

The correctness of the implementation is verified. To test the performance of the implementation we use as a reference the existing Secure Extension to the OLSR protocol[7]

In table I we provide some measures from an emulation performed using the Netkit [11] network emulator running on a computer with an AMD64 2GHz CPU.

The "Symmetric" column refers to [7], which uses 128 bit long keys, which is a good trade-off between performance and security using symmetric cryptography. In order to provide a similar level of security with asymmetric cryptography, to our advice at least 1024 bits long keys should be employed.

As we expected the per task processing time is much higher, but in respect to the OLSR protocol times, the nodes do achieve to discover the topology of the network and route correctly the packets.

VI. Conclusions and future work

In this paper we presented a novel approach to secure routing control traffic in wireless mesh network. The solution we propose

References

[1] A. Barbir, S. Murphy, and Y. Yang, "Generic threats to routing protocols," IETF RFC 4593, October 2006. [Online]. Available: http://tools.ietf.org/html/rfc4593

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

[3] K. Weniger, "PACMAN: Passive autoconfiguration for mobile ad hoc networks," *IEEE Journal on selected areas in communications*, vol. 23, no. 3, pp. 507–519, March 2005.

[4] J. Eriksson, M. Faloutsos, and S. V. Krishnamurthy, "Dart: Dynamic address routing for scalable ad hoc and mesh networks," *Networking, IEEE/ACM Transactions on*, vol. 15, no. 1, pp. 119–132, 2007. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4100713

[5] Freaknet Medialab, "Netsukuku – Close the world, ʇxǝu ǝɥʇ uǝdO," July 2008. [Online]. Available: http://netsukuku.freaknet.org/2html/documentation/main_documentation/netsukuku.pdf

[6] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," IETF RFC 3626, October 2003. [Online]. Available: http://tools.ietf.org/html/rfc3626

[7] A. Hafslund, A. Tønnesen, R. B. Rotvik, J. Andersson, and O. Kure, "Secure extension to the OLSR protocol," in *OLSR Interop and Workshop*, August 2004.

[8] P. R. Zimmermann, *The official PGP user's guide*. Cambridge, MA, USA: MIT Press, 1995.

[9] A. Tønnesen, "Implementing and extending the optimized link state routing protocol," Master's thesis, UniK University Graduate Center - University of Oslo, 2004.

[10] "The GNU privacy guard," http://www.gnupg.org/, 2008.

[11] M. Rimondini, "Emulation of computer networks with netkit," Dipartimento di Informatica e Automazione, Roma Tre University, Tech. Rep. RT-DIA-113-2007, January 2007. [Online]. Available: http://www.netkit.org/

Bibliography

[1] Cédric Adjih, Daniele Raffo, and Paul Mühlethaler. Attacks against OLSR: Distributed key management for security. In *2005 OLSR Interop and Workshop*, Ecole Polytechnique, Palaiseau, France, July 28–29 2005.

[2] Andreas Hafslund, Andreas Tønnesen, Roar Bjørgum Rotvik, Jon Andersson, and Øivind Kure. Secure extension to the OLSR protocol. In *OLSR Interop and Workshop*, August 2004.

[3] T. Dierks and E. Rescorla. The transport layer security (TLS) protocol version 1.1. IETF RFC 4346, April 2006.

[4] Cedric Adjih, Thomas Clausen, Philippe Jacquet, Anis Laouiti, Paul Mühlethaler, and Daniele Raffo. Securing the OLSR protocol. In *2nd IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net 2003), Mahdia*, pages 25–27, 2003.

[5] Bruce Schneier. Attack trees – modeling security threats. *Dr Dobb's Journal*, (n.12), December 1999.

[6] Andreas Tønnesen. Implementing and extending the optimized link state routing protocol. Master's thesis, UniK University Graduate Center - University of Oslo, 2004.

[7] Free Software Foundation, Inc. The GNU general public license. `http://www.gnu.org/licenses/gpl.html`.

[8] The BSD license. http://www.opensource.org/licenses/bsd-license.php.

[9] Olsr.org website. http://www.olsr.org/.

[10] Olsr-ng. http://olsr.funkfeuer.at/.

[11] OpenSSL: The open source toolkit for SSL/TLS. http://www.openssl.org/.

[12] Announcing the secure hash standard. NIST Federal Information Processing Standards Publication 180-2 (+ Change Notice to include SHA-224), August 2002.

[13] Ron Rivest. The MD5 message-digest algorithm. IETF RFC 1321, April 1992.

[14] Seattle wireless. http://seattlewireless.net/.

[15] Guifi. http://www.guifi.net/.

[16] Metrix Communication LLC. http://www.metrix.net/.

[17] Nepal wireless. http://www.nepalwireless.net/.

[18] Seán Ó Siochrú and Bruce Girard. *Community-based Networks and Innovative Technologies: New Models to Serve and Empower the Poor.* United Nations Development Programme, 2005.

[19] Djurslands international institute of rural wireless broadband. http://www.diirwb.net/.

[20] Wireless ghana. http://www.wirelessghana.com/.

[21] Athens wireless metropolitan network. http://www.awmn.net.

[22] Freifunk. http://www.freifunk.net/.

[23] Funkfeuer. http://funkfeuer.at/.

[24] Ninux.org wireless community network. http://ninux.org/.

[25] Copyriot community. http://www.copy-riot.org/.

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

[27] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich. Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.). IETF Internet-Draft draft-openmesh-b-a-t-m-a-n-00, March 2008.

[28] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (bgp-4). IETF RFC 4271, January 2006.

[29] Pyramid linux. http://pyramid.metrix.net.

[30] OpenWRT. http://www.openwrt.org/.

[31] Olsrd link quality extensions. http://www.olsr.org/docs/README-Link-Quality.html.

[32] B.A.T.M.A.N. - Better Approach to Mobile Ad-hoc Networking. http://www.open-mesh.net/batman.

[33] madwifi.org. http://www.madwifi.org/.

[34] Pico peering agreement v1.0. http://www.picopeer.net/PPA-en.html.

[35] Freenetworks.org peering agreement v1.1. http://freenetworks.org/peering.html.

[36] NYCwireless. http://www.nycwireless.net/.

[37] Fon. http://www.fon.com/.

[38] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP message format. IETF RFC 4880, November 2007.

[39] Philip R. Zimmermann. *The official PGP user's guide.* MIT Press, Cambridge, MA, USA, 1995.

[40] The GNU privacy guard. `http://www.gnupg.org/`.

[41] The GNU operating system. `http://www.gnu.org/`.

[42] Philip R. Zimmermann et al. *An Introduction to Cryptography.* PGP 6.0 Documentation, 1998.

[43] Marc Horowitz. A pgp public key server. Master's thesis, Massachusetts Institute of Technology, 1996.

[44] The Linux Kernel Archives. `http://www.kernel.org/`.

[45] Matthew Marsh. *Policy Routing Using Linux.* Sams, March 2001.

[46] Iproute2 - The Linux Foundation. `http://www.linuxfoundation.org/en/Net:Iproute2`.

[47] iptables. `http://www.netfilter.org/`.

[48] Massimo Rimondini. Emulation of computer networks with netkit. Technical Report RT-DIA-113-2007, Dipartimento di Informatica e Automazione, Roma Tre University, January 2007.

[49] Netkit – the poor man's system to experiment computer networking. `http://www.netkit.org/`.

[50] Claudio Pisa. Web of trust plugin patch to the olsrd daemon. `http://stud.netgroup.uniroma2.it/~claudio.pisa/wotplugin/addwot-olsrd-0.5.6-r2.patch`.

[51] IEEE international conference on communications 2009, june 14-18 dresden, germany. `http://www.ieee-icc.org/2009/`.

[52] Security and communication networks journal. `http://www3.interscience.wiley.com/journal/114299116/home`.