Computer Communications xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Trust-based security for the OLSR routing protocol

Asma Adnane [a,*], Christophe Bidan [b], Rafael Timóteo de Sousa Júnior [c]

[a] School of Computing and Mathematics, University of Derby, United Kingdom
[b] SUPELEC, SSIR Team, Avenue de la Boulaie, Cesson-Sévigné 35510, France
[c] Electrical Engineering Department, University of Braslia, Braslia, DF 70910-900, Brazil

ARTICLE INFO

Article history:
Received 23 March 2011
Received in revised form 1 November 2012
Accepted 3 April 2013
Available online xxxx

Keywords:
Trust
Ad hoc networks
Security
Routing protocol
OLSR

ABSTRACT

The trust is always present implicitly in the protocols based on cooperation, in particular, between the entities involved in routing operations in Ad hoc networks. Indeed, as the wireless range of such nodes is limited, the nodes mutually cooperate with their neighbors in order to extend the remote nodes and the entire network. In our work, we are interested by trust as security solution for OLSR protocol. This approach fits particularly with characteristics of ad hoc networks. Moreover, the explicit trust management allows entities to reason with and about trust, and to take decisions regarding other entities.

In this paper, we detail the techniques and the contributions in trust-based security in OLSR. We present trust-based analysis of the OLSR protocol using trust specification language, and we show how trust-based reasoning can allow each node to evaluate the behavior of the other nodes. After the detection of misbehaving nodes, we propose solutions of prevention and countermeasures to resolve the situations of inconsistency, and counter the malicious nodes. We demonstrate the effectiveness of our solution taking different simulated attacks scenarios. Our approach brings few modifications and is still compatible with the bare OLSR.

1. Introduction

Today, mobile Ad-hoc networks (MANETs) are a major element of the business environment, allowing wireless devices such as cell phones, laptops, and PDAs to provide mobility to users and enable them to keep in constant contact with others. Technically, MANETs are self-organized wireless mobile networks that do not rely on any centralized administration or fixed network infrastructure. The cooperation between the mobile devices allows to provide the network services. More precisely, each device participates in routing service: a communication between distant devices can be established only if intermediate devices cooperate by forwarding the messages they receive. Thus, each device of a MANET has to maintain a local routing table that determines the next hop toward all other devices. The routing table is managed using an *ad hoc routing protocol* (for example: OLSR, AODV).

Many ad hoc routing protocols have been developed for ad hoc networks [1]. Roughly speaking, they can be classified according to the type of route discovery: reactive and proactive. In reactive protocols, e.g. AODV (Ad hoc On-demand Distance Vector), the routing request is sent on-demand: if a device wants to communicate with another, then it broadcasts a route request and expects a response from the destination. Conversely, proactive protocols update their routing information continuously in order to have a permanent overview of the network topology (e.g. OLSR [2]).

The security of MANET is a major challenge, and the self organization characteristics of MANET imply that traditional security solutions are often inadequate. In other words, any device participating to the routing service can easily attack the MANET either by disrupting any communication with which it is involved, or by compromising the routing tables of other devices. It is important to point out that these two attacks affect the network at two different levels: the first one is the message routing, whereas the second is the ad hoc routing protocol.

As regards the security of the message routing, the classical approach consists in using reputation systems to detect misbehavioral devices (e.g. devices that do not forward the messages). Concerning the security of the ad hoc routing protocol, most research assumes that as long as the messages containing the topological information are secured, the routing tables cannot be compromised. Our point of view is that such an approach is not sufficient since in any ad hoc routing protocol, a device can easily compromise the routing tables by sending incorrect topological information in secured messages. Thus, solutions that guarantee the correctness of the routing tables have to be proposed.

Assuming that any protocol is based on implicit trust relations (as demonstrated in [3] and Section 4), we assert that such trust relations can be used by each device to assess the expected correct

* Corresponding author.
E-mail addresses: a.adnane@derby.ac.uk (A. Adnane), christophe.bidan@supelec.fr (C. Bidan), desousa@unb.br (R.T. de Sousa Júnior).

0140-3664/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comcom.2013.04.003

behavior of the other devices, and also to reason about the correctness of its routing table. In this article, we illustrate this through the OLSR (Optimized Link State Routing protocol [2]) protocol. We summarize our contributions to the analysis of the implicit trust within OLSR, and to the trust-based reasoning and countermeasures for securing OLSR nodes.

The paper is organized as follows. In Section 2, related works on security in ad hoc networks are summarized. In Section 3, we introduce the concept of trust management and trust specification language. An overview of OLSR is presented in Section 4. In Section 5, we introduce the analysis of implicit trust in OLSR, then we present trust reasoning developed to secure OLSR in Section 6. Countermeasures concerning the attacks against the basic operations in OLSR, and a method of distribution of information about trust relation to prove the attack and prevent distant nodes in the network are detailed in Section 7. Finally, we conclude this paper by presenting simulation results and our future works.

2. Related works

As we pointed out before, the routing service in MANET can be attacked either by disrupting the message routing or by compromising the routing tables. In the former case, the main concern is to protect against misbehaving devices, and especially selfish devices (i.e. devices that do not properly forward messages). The traditional solution consists in forcing the devices to collaborate. One of the early works on collaboration is presented by Marti et al. [4]. The authors introduce the ***watchdog*** and ***pathrater*** mechanisms. Basically, the *watchdog* mechanism is used by each node to monitor the behavior of its neighbors. Using the information of the *watchdog*, the device can locally compute a rating for each of its neighbor, and when this rating is below a given threshold, it uses the *pathrater* mechanism to compute another path avoiding misbehaving devices. Thus, selfish devices are detected and not used anymore. Note that Marti et al. do not allow each device to notify other devices when a malicious device is detected.

Today, a major part of the research works on collaboration in MANET has been inspired by this previous work, especially by using the *watchdog* mechanism to build a reputation system [5,6,8–10]. For example, in [6], the authors propose a collaboration system called CONFIDANT (Cooperation Of Nodes: Fairness In Dynamic Ad-hoc NeTworks) based on a reputation system. More specifically, each device uses a *watchdog* mechanism to monitor and report the message routing behavior of other devices. Given the observed and reported behavior, each device computes a reputation score for each device to detect misbehaving ones. The detection of misbehaving devices leads to their isolation. Thus, the devices are constrained to cooperate so as not to be isolated. Later, the same authors extended their cooperation system in order to deal with devices that deliberately send false reputation scores [11].

Similarly, Michiardi et al. propose a cooperation enforcement mechanism, called CORE (COllaborative REputation) [7]. Basically, CORE uses a *watchdog* mechanism to allow each device to monitor its neighbors. Based on its own observation as well as the scores provided by other devices involved in the current operation, a device can compute a reputation score for each of its neighbor, this score represents the degree of cooperation. Then, when a selfish device is detected, it is gradually denied network services. Thus, a device cooperates, otherwise it can no longer use the MANET. Notice that CORE allows to rehabilitate selfish devices if they behave correctly again.

In contrast, *Buttyan and Hubaux* have proposed the collaboration mechanism, called Nuglets [12] adopting a completely different approach. They introduce a virtual currency called *nuglet*. Each node has to pay to use network services (forwarding its data), and must be paid for offering services to other nodes. Thus, selfish nodes will finish their nuglets and can no longer send packets. The drawback of this method is that the nuglets are managed by a centralized entity.

In brief, collaboration systems are based either on reputation systems monitoring the neighbors' behavior to detect misbehaving devices (e.g. selfish nodes), or on a virtual currency to enforce the nodes to collaborate. However, in both cases, the solutions implicitly assume that the routing tables are correct.

However, other works propose reputation systems that can also be applied for securing routing protocol by monitoring node behavior, in order to verify that nodes respect the routing protocol specification. For example, CORE [7] is suggested as a generic mechanism and can be integrated with any network service. Precisely, Meka et al. [10] propose trust-based reputation model for AODV. Reputation is calculated according to the degree of participation in the routing protocol and the information it provides about the network topology. Reputation system was also used as a security method to perform trust-based multi-path routing [13,14]. Thus, they do not allow to detect a malicious node which would be a normal behavior in terms of message routing, but a misbehavior with respect to the ad hoc routing protocol.

To deal with compromised routing tables, the major part of the research works is based on cryptography to secure the messages containing the topology information used to calculate the routing tables. The underlying assumption is that authenticated devices are known to behave correctly: when some topology information is authenticated, it is correct/trusted. In other words, the main concern of these research works is to keep *unknown* devices (i.e. intruders) out of the MANET, thus stopping such devices from modifying/falsifying topology information sent by authenticated devices.

Many research works have proposed security solutions for reactive ad hoc routing protocol based on cryptography [15–17]. For example, Zapata and Asokan [16] have proposed a secured version of AODV. The authors present two mechanisms to secure the AODV messages: digital signatures to authenticate the non-mutable fields of the messages, and hash chains to secure the hop count information (the only mutable information in the messages). Ariadne [17] is another secured protocol based on DSR and TESLA: the authors assume that a shared secret key is distributed for a group of trusted nodes using TESLA and that the nodes are synchronized.

Similarly, some research works have been undertaken for the security of proactive ad hoc routing protocols based on cryptography [19,20]. For example, Adjih et al. [20] have proposed a secured version of OLSR called SOLSR. Their approach is based on the signature and time-stamp of each OLSR control message. A signature is generated for each control message and sent with the message to prevent malicious nodes to modify or falsify topology information. In addition, a time-stamp is associated with each signature to estimate the freshness of the message. However, This solution does not ensure the correctness of the information provided by authenticated nodes, and assumes that any authenticated node is a trusted node without any verification. The solution of Hafslunf et al. consists in signing the OLSR packets. In our view, this latter approach is not adapted since a corrupted node can easily modify the content of a *TC* message before generating the signature of the new packet which will contain it.

Omar et al. [18] propose a fully distributed public key certificate management system based on trust graphs and threshold cryptography. It allows nodes to issue public key certificates, and to authenticate the other nodes via certificates chains without trusted authority. The proposed solution uses the threshold cryptography to resist against false public keys certification. The initialization

Q1 Please cite this article in press as: A. Adnane et al., Trust-based security for the OLSR routing protocol, Comput. Commun. (2013), http://dx.doi.org/10.1016/j.comcom.2013.04.003

phase is based on Shamir's secret, the secret key is kept secret at the system dealer which is a telecommunication service provider common to all current members. The authors assume that trust relationships exists between initial nodes. So, to join the network, the new node have to provide a trust evidence which is authenticated by a member node.

In brief, solutions based on cryptography implicitly assume that an *authenticated* node is trustworthy and that the topology information it generates is correct. But these solutions do not allow to prevent a *malicious authenticated* device to disseminate incorrect topological information. However, we can note that Raffo et al. [21] have extended SOLSR to address the problem where *authenticated* nodes have been compromised by attackers, and can then inject false (though correctly signed) topology messages. The basic idea is that each node signs its topology messages, these signatures being reused later by other devices to prove their own topology messages. The resulting solution allows to prevent devices to declare imaginary links. However, it does not allow to verify the correctness of the routing tables.

Another topic of research concerns intrusion detection. Briefly speaking, an intrusion detection system (IDS) uses a *watchdog* mechanism to monitor some events, and analyzes those events to find malicious devices. This analysis consists in either comparing the current behavior with the *correct* behavior (i.e. anomaly detection), or finding attack scenarios from a signature database (i.e. misuse detection). In the context of MANETs, most of the solutions propose a generic distributed architecture, each node having its own local intrusion detection system (LIDS), and global detection being performed thanks to a module that allows the cooperation between the LIDS [22,23]. The monitored events are generally the neighbor behaviors, any misbehaviors being an intrusion.

Wang et al. [24] and Cuppens-Boulahia et al. [25] were interested in securing OLSR protocol by using semantic properties of the protocol. They propose an intrusion detection system for verifying the correctness of OLSR control messages.

Wang et al. [24] present an intrusion detection approach for OLSR based on checking protocol semantics (messages consistency). The semantic properties that are implied by the protocol definition are used by every MANET node for conflict checking regarding the correct OLSR routing behavior. We show hereafter that these properties can be expressed in terms of trust, allowing the formulation of a basic intrusion detection mechanism based on mistrust. Moreover, the reasoning on trust is shown to be useful for a node to counter other attacks that circumvent these semantic properties. However, the authors do not propose countermeasures to stop and isolate malicious nodes.

Cuppens-Boulahia et al. [25] have precisely sought to define such countermeasures against malicious nodes. Based on the same detection mechanism as that of Wang et al. [24], the authors analyze the possible attacks against OLSR, and present solutions to counter attacks by offering legitimate nodes new routes that do not include malicious nodes. However, detection is again limited to the local vision and does not check the consistency of all these observations (for example, they do not check the consistency of Topology Control (TC) messages and the routing table [2]). On the other hand, the authors do not verify that the new calculated routes are consistent with the specification of OLSR, and thus they do not ensure that it will not be detected as an inconsistency.

To our knowledge, only Wang et al. [24] and Cuppens-Boulahia et al. [25] were interested in semantic properties in OLSR protocol. Their approach is presented as an intrusion detection system. In our view, it can be regarded as a trust reasoning because it allows each node to reason and to verify the consistency of received information before trusting the sender. However, their approach is limited to semantic properties of ***local observation***, and does not allow sharing local observations.

In our work, we assume that it is important to define the correct behavior in the routing operation to detect attacks against the OLSR protocol. We first analyze the semantics properties of OLSR in terms of trust [26], and we identify implicit trust-related properties in OLSR that can be used for malicious purpose. Then we propose to integrate trust-based reasoning in each OLSR node for the verification of ***local*** and ***global*** vision of the network [27–29]. Finally, we present two measures to counter attacks against OLSR: prevention that solves some protocol vulnerabilities, and countermeasures that deal with misbehavior and inconsistency because prevention measures cannot solve all the vulnerabilities causing inconsistency. The proposed prevention measures are based on message signature proposed by Adjih et al. [20] (that we call *SOLSR*). The difference is that we use the provable identity mechanism instead of a trusted centralized authority for public key certification. The resulting mechanisms allow to resolve the OLSR vulnerabilities which are due to the easy usurpation of node identity and the lack of links verification at the neighborhood discovery. However, these mechanisms do not resolve other vulnerabilities, such as the lack of monitoring of the establishment of the routing tables. Thus, when other vulnerabilities are exploited and an attack is detected, we propose countermeasures to isolate malicious nodes.

To sum up, routing service in MANET is provided by an ad hoc routing protocol that allows each device to calculate and maintain its routing tables, and the routing messages using the resulting routing tables. Cryptographic mechanisms allow to secure the exchanges between the devices, and in particular, the topology information used to calculate the routing tables, whereas cooperation and reputation mechanisms can contribute to enforce the devices to participate in the routing messages. However, none of these approaches allow to guarantee the correctness of the routing tables. Thus, we assert that these approaches should be reinforced by a specific mechanism that permits each device to verify that its routing table is correct, i.e. it has a correct vision of the network topology.

In our contribution, we propose for each device to assess the behavior of the other devices with respect to the implicit trust relations induced by the OLSR protocol. Note that since we focus on malicious devices that disseminate topological information to compromise the routing tables, we suppose that SOLSR [20] is used to ensure the security of the OLSR messages. SOLSR ensures only the security of OLSR messages, but does not ensures that the source authenticated node is not giving wrong topology information. Our work verify that trusted nodes (authenticated nodes) are behaving correctly, and respecting the OLSR specification. Trust is build with correct behavior, and not created only with authentication.

3. Trust management in ad hoc routing

Trust, trust models and trust management have been the subjects of several ongoing research projects. Trust is recognized as an important aspect for decision-making in distributed and auto-organized applications [3].

In the literature, there is no consensus on the definition of trust and what trust management encompasses. Many authors propose their own definitions of trust, each one concerning a specific research area such as authentication [31], e-commerce, P2P, and many other fields. As a result, a multitude of formal models for trust calculation and management have emerged. However, this also leads to a certain conceptual confusion, indicated by the fact that similar concepts appear under different names and reciprocally. To avoid this confusion, in the present paper, we use the trust definition and a language to express trust proposed by Yahalom et al. [31] which together allow to formalize and clarify trust aspects in communication protocols:

Q1 Please cite this article in press as: A. Adnane et al., Trust-based security for the OLSR routing protocol, Comput. Commun. (2013), http://dx.doi.org/10.1016/j.comcom.2013.04.003

"The notion of trust is fundamental for understanding the interactions between devices such as human beings, organizations, nations and others. The fact that a node A trusts a node B in some respect, informally, means that A believes that B will behave in a certain way and will perform some action under certain specific circumstances".

3.1. Trust classes

Trust relation is established according to the possibility of conducting a protocol operation (action), and is evaluated by the entity A on the basis of what it knows about the entity B and the circumstances of the operation. According to the action undertaken and its circumstances of execution, it is necessary to distinguish various trust classes as defined in [31,32]. For the sake of precision on the formalization of trust relations required by OLSR, we propose appropriate classes for the actions performed by this protocol (i.e, the trust in another entity to route messages: routing trust). First, we use trust classes defined in [31] in the context of the analysis of authentication protocols. An entity may trust another one for one of the following actions:

- Providing identification of one entity to another (the class *identification*: id).
- Not interfering in other entities sessions, neither passively by reading other secret messages, nor actively by impersonating one of the parties (class *not-interfering*: ni).
- Providing trust recommendations about other entities (rec).

Then, we also use the classes proposed in [32], specifically:

- A trusting entity which grants the access to its resources or services by other entities (*access trust* class: at).
- A truster trusts a trustee to make decisions on its behalf, with respect to a resource or service that the truster owns or controls (*delegation trust* class: dt).

Finally, we define sub-classes for actions carried out effectively by OLSR protocol, such as trust for routing (*fw*).

3.2. Trust specification language

We use the language proposed by Yahalom et al. [31] for expressing the clauses concerning trust in a networking protocol. The trust relation is taken into account if the possibility of realization of a protocol operation (the action) is evaluated by entity A on the basis of what it knows about entity B and the circumstances of this operation.

We distinguish the direct trust relations and the derived trust relations as mentioned in [31], the latter being established on recommendations from other entities. Given the presence of several types of entities in the execution environment of a protocol, and the existence of indirect relationship between the entities, it is necessary to distinguish these two types of trust relations. Thus, the clauses relating to trust are expressed with the following notations:

- Each entity is identified by a single name; the terms A, B, C indicate specific entities.
- A specific trust class is noted cc.
- The formula $A\ trusts_{cc}(B)$ means that A trusts B with respect to the action cc.
- $A\ trusts_{cc}(S)$ means that A trusts the set of entities S with respect to action cc, S being defined as the set of all entities for which a certain predicate holds.
- $A\ trusts_{cc-C}(B)$ means that A trusts B to perform action cc with respect to the entity C (but not necessarily to other entities).
- $A\ trusts.rec_{cc}(B)\ when.path[S]\ when.target[R]$ means that A trusts the recommendations of entity B about the capacity of other entities to perform action cc. The *when* clauses allow the specification of constraints on the recommendations. The trust recommendation *path* is a sequence of entities such that each one is recommended by its predecessor. So, the *when.path* specifies the only set of entities to be considered as candidates for the next step in the recommendation path. The *target* clauses specifies the only set of entities to be considered as candidates for becoming target entities in some recommendation paths.
- $A\ trusts.rec^{*}_{cc-C}(B)$ presents the derived recommendations deduced from previous recommendations of B.
- $A\neg trusts(B)$ means that A mistrusts B. In our approach, we consider that mistrust relation is total and does not concern a particular action. If B is compromised (i.e. it does not respect the routing protocol), and does not complete requested action, then A mistrusts it for all other actions. We call this relation ***exact*** mistrust.
- $A\neg trusts(S)$ means that A mistrusts all the nodes included in S. This mistrust relationship is called ***partial*** mistrust, because A detects an inconsistency in the achievement of the action requested from the group of nodes S, but is unable to identify precisely which node. It is important to mention that the notation $A\neg trusts(\{B,C\})$ is not equivalent to $A\neg trusts(B)$ and $A\neg trusts(C) : A\neg trusts(\{B,C\})$ means that A detects an inconsistency between nodes B and C, but is unable to identify precisely which nodes are malicious. The mistrust relation is related to the action accomplished by a group of nodes.

In the following sections, the use of this language, together with the mathematical set theory, allows to reason about the trust required by the OLSR protocol and to explicitly express trust relations between the nodes executing this protocol.

4. OLSR protocol: overview and trust-based analysis

OLSR is a proactive link-state routing protocol, which uses an optimized flooding mechanism to broadcast partial link state information to all network nodes. The protocol uses multi-point relays (MPR) which are selected nodes that forward broadcasted messages during the flooding process. The link state information is generated only by nodes elected as MPRs, and each MPR must only report on the state of links between itself and its selectors. Two types of control messages, HELLO and TC (Topological Control), allow each node to obtain and declare network topological information. HELLO and TC messages have validity time, which indicates for how long time after reception a node must consider the information contained in the message as valid, unless a more recent update to the information is received.

The functioning of OLSR occurs in three steps: neighborhood discovery, MPR selection, and Routing table calculation. In this section, while presenting the OLSR protocol, we analyze the implicit trust relationships established between OLSR nodes. This analysis allows the identification of trust assumptions built during the different steps of the protocol and how it can be used by attackers.

4.1. Notations

In OLSR, each node maintains its local vision of the network. This vision consists of the following sets:

- *MANET*: the set of the whole MANET nodes,

- LS_x: the link set of node $x : LS_x = \{MANET, \{asym, sym\}\}$. It includes the set of asymmetric (status = *asym*) and symmetric neighbors (status = *sym*) of node *x*.
- NS_x: the set of symmetric neighbors of node *x*,
- $2HNS_x$: the set of two-hop neighbors of node *x*,
- $MPRS_x$: the set of nodes selected as MPRs by node *x* ($MPRS_x \subseteq NS_x$), which are in charge of routing and forwarding the packets sent by *x*.
- MSS_x (MPR Selection Set): the set of symmetric neighbors which have selected node *x* as MPR ($MSS_x \subseteq NS_x$).
- RT_x (Routing Table): the routing table of node *x*.
- $route_{x \to y}$ is the sequence of nodes included in the route (shortest path) between *x* and *y*, which has the form of the following predicate:
  $route_{y_1 \to y_n} = y_1, \ldots, y_n$, *where* $y_{i+1} \in MPRS_{y_i}$.
- *TopologySet*: the set of information provided by the received TC message, which describes the network topology.

Each node collects information about the network by exchanging *HELLO* and *TC* messages. These exchanges are presented by the following formula:

- $HELLO_x$ is the *HELLO* message generated by node *x*; it includes the set of the neighbors of *x*. This message is diffused periodically and is not broadcasted. In this message, each neighbor is presented with the following formula: $HELLO_x = \{MANET, \{asym, sym, mpr\}\}$ Thus, the HELLO message is used to deduce the sets LS_x, NS_x and $MPRS_x$. As HELLO messages are diffused periodically by *x*, LS_x, NS_x and $MPRS_x$ can be deduced by all the neighbors of *x*.
- TC_x is the TC message generated by node *x* and contains the list of its MPR selectors $MSS : TC_x = MSS_x$. In OLSR, TC messages are diffused periodically, and broadcasted only by the MPR nodes.
- $x \overset{HELLO_y}{\leftarrow} y$, $x \overset{TC_y}{\leftarrow} y$: respectively, the reception of *HELLO* and *TC* messages from *y* by node *x*.
- $x \overset{TC_x}{\rightarrow} *$, $x \overset{DATA_x}{\rightarrow} * : x$ broadcasts its *TC* and *DATA* messages. These messages should be retransmitted by the MPR nodes of *x*.
- $(TC_x)_y$: this notation signifies the TC message of *x* is retransmitted by *y*.
- $x \overset{TC_y}{\nleftarrow} y$, $x \overset{DATA_x}{\nleftarrow} y$: when *y* is MPR of *x*, this notation means *y* does not retransmit the data/TC messages generated by *x* after an awaiting period.

OLSR does not implement this event. In our work, we have defined an event that waits for a specified period after sending each TC and DATA messages, the event waits to receive the same message from the MPR nodes of *x* (MPR nodes have to forward messages of their selectors). If the waiting time elapses, and the message is not received, the event returns an error.

4.2. Neighborhood discovery

The first step in OLSR is neighborhood discovery, it allows each node to detect one-hop and two-hop neighbors by exchanging HELLO messages. HELLO messages are sent periodically by a node to advertise its links (declared as asymmetric, symmetric or MPR) with neighbor nodes. Received HELLO messages allow a node to memorize information about links and nodes within its two-hop neighborhood, and to constitute the internal local state of each node which is represented under the form of sets, (LS, NS, 2HNS, MPRS, MSS). These sets are updated and used continuously for MPR selection, in such way that a message sent by the node and relayed by its MPR set (i.e. the elements of its MPRS) will be received by all its two-hop neighbors. Each node also records the addresses of its neighbors that selected it as MPR (what constitutes the MSS). Thus, HELLO messages allow a node to establish its view of the "small world" (up to the two-hop neighborhood).

Given the general description of the protocol and the definition of the sets maintained by the OLSR node, the language described in Section 3.2 may be used to express the trust relationships in this protocol. In OLSR, the nodes are considered to be cooperative and to trust the fact of obtaining the cooperation of the neighbor nodes. This behavior corresponds to the concept of general trust as defined in [3]. For example, the RFC 3626 [2] states that "a node should always use the same address as its main address" (p. 5), which is the basic belief of a node in the identity of others. This statement is translated using the formal language into the following formula:

$$\forall A, B \in MANET : Atrusts_{id}(B)$$

Initially, a node *A* does not have any view of the network (no neighbors). The node starts building its view with the reception of HELLO messages coming from the neighbors. These messages allow the node to detect asymmetrical links, leading to a modification of the local state of *A* about its trust in a neighbor node *B*, i.e. *A* knows *B* but does not have opinion about it yet, because *A* is not sure that *B* respects the OLSR specification with regard to the reception and sending of HELLO messages: $A \overset{HELLO_B}{\leftarrow} B : (A, sym) \notin LS_B \; or (A, asym) \notin LS_B \Rightarrow LS_A \leftarrow LS_A \cup (B, asym)$.

This formula means that *A* does not have opinion about *B*, although *A* receives HELLO messages from *B*. However, being an agent generally trustful [3], *A* broadcasts HELLO messages that can be received by *B* which in turn will be able to take them into account, and then adds *A* to its set of symmetrical neighbors NS_B (Fig. 1). If *B* acts according to the protocol, i.e. it sends HELLO messages advertising a link with *A*, then a new situation of trust is reached:

$$\begin{aligned} A \overset{HELLO_B}{\leftarrow} B, (A, link\_state) \in LS_B, link\_state \in \{asym, sym\} \\ \Rightarrow A \; trusts_{id \cup ni}(B), \\ LS_A = LS_A \cup (B, sym), \\ NS_A = NS_A \cup \{B\}, \\ 2HNS_A = 2HNS_A \cup (NS_B - \{A\}) \end{aligned} \quad (1)$$

A trust relation has just been built which is concretized by the fact that now *A* regards *B* as its symmetrical neighbor, and the symmetrical neighbors of *B* as 2-hop neighbors. In addition, this trust relation is seen as symmetrical, since *B* is expected to behave in the same way as *A*:

$$B \overset{HELLO}{\leftarrow} A \Rightarrow B \; trusts_{id \cup ni}(A)$$

This symmetrical relation is the basis for future decisions which will be taken by *A* about its small world (MPR selection), but also, for the routing towards the large world (calculation of the routing table)

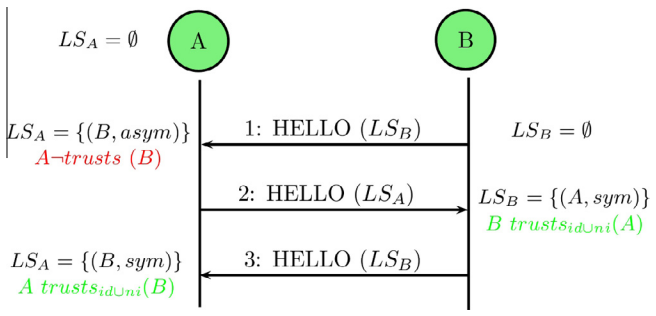


Fig. 1. Trust establishment during neighborhood discovery between *A* and *B*.

through the exchange of TC messages. The following figure sums up trust relations during neighborhood discovery Fig. 1:

4.3. MPR selection

MPR selection is the most important operation in OLSR. After the detection of one-hop and two-hop neighbors, each node must, among its one-hop neighbors, select the minimum number of nodes enabling it to reach all the two-hop neighbors. The selected neighbors are called MPRs, and are advertised in the HELLO message with "mpr" status.

For example, in Fig. 2, node A ($NS_A = \{B, C, E\}, 2HNS_A = \{D, F\}$) has to select C as MPR, because it provides reachability to F. As D can also be reachable via C, then C will be the only MPR of A.

When a node is selected as MPR, it adds the selectors in its *MSS* and generates a TC message advertising all these nodes as illustrated in Fig. 3.

TC messages contain the topological information necessary for computing routes to the whole network, the "big world". The reception of TC messages allows a node to obtain information about destination nodes and to keep this information in its TopologySet. Each node selected as MPR periodically broadcasts TC messages advertising its MPR selectors neighbors. TC messages are flooded in the whole network allowing the nodes to compute the network topology to be used for routing (routing table). In terms of trust, when a node A selects a node C to be MPR (router), this means A trusts C (and all the nodes in its *MPRS*) for routing:

$$\forall y \in MPRS_A : A\ trusts_{fw}(y) \tag{2}$$

The MPR nodes of A also have to select their MPRs and so on. Consequently, the MPR nodes of A are required to recommend A the routes to the distant nodes. So, each node trusts the recommendation of its MPRs for routing to any distant nodes:

$$\forall z \in MANET,\ \exists y \in MPR_A\ A\ trusts.rec_{fw}\ (y) when.path[route_{y\to z}]\ when.target[z] \tag{3}$$

This formula presents the general rule of trust recursiveness for the routing in the networks operating under OLSR. Indeed, the routing success depends not only on the local MPR selection, but also on the MPR selection of neighbors. Therefore, each node must trusts the MPR selection of its MPR and so on. On the reception of HELLO messages, the MPR node C discovers that it is selected as MPR by A (link with mpr status) and updates its *MSS*. This calculation allows a node C to discover information about the trust that other nodes place in it.

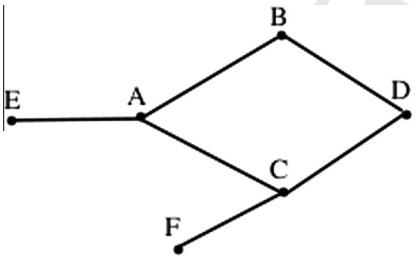


Fig. 2. Example.

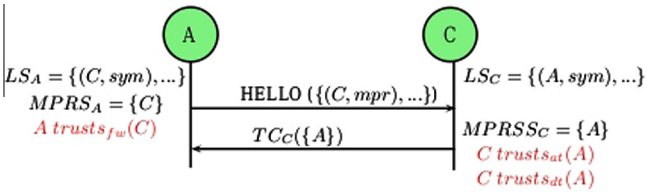


Fig. 3. Trust establishment during MPR selection: C is MPR of A.

As C allows the nodes of its *MSS* to use its resources for routing, which constitutes a form of access trust as discussed in Section 3, the calculation of MSS_C implies the following trust relations (Fig. 3):

- C trusts A to use its resources for routing without causing prejudice: $C\ trusts_{at}(A)$
- C trusts A for advertising that C is its MPR: $C\ trusts_{dt}\ (A)$

4.4. Routing table calculation

The routing table is the result of the OLSR protocol. Each node creates its point of view of the network topology, and calculates the shortest path to any destination using the ***Dijkstra*** shortest path algorithm. The routing table *RT* is described by the following formula: $\forall z \in MANET,\ \exists y \in MPRS_x \Rightarrow \exists T \in RT_x,\ T = (z, y, N, I)$. Each entry in *RT* consists of (z, y, N, I), and specifies that the node identified by z is located N hops away from the local node. The symmetric neighbor node which is identified by y is the next hop node in the route to z. This symmetric neighbor node is reachable through the local interface I.

From the trust point of view, the calculation of the shortest path between x and z through the MPR y means that x trusts y for the routing towards z. If we note $T = (z, y, N, I)$, a tuple of RT_x, the following relation is obtained:

$$\forall T \in RT_x \ \Rightarrow\ x\ trusts_{fw-z}(y) \tag{4}$$

Moreover, the routing table is calculated in order to have only one route towards each destination, and each selected route is the shortest among the routes starting from MPR nodes. The inherent risk in the choice of only one route towards any destination is to choose a misbehaving node as a router. In [26], we explain how this vulnerability can be exploited by attackers, who give false information about the network topology in order to redirect all the traffic of the network towards them and/or to disturb protocol operation.

Even if there are several paths towards z, x will only choose the shortest route starting from one of its MPRs. The routing table calculation is a reasoning based on the distance and results in the set of routes which the node considers as the most adequate for the routing. After computing the distances to each destination, the node will place more trust on the nodes which offer the shortest path (formula 4).

The selection of y as MPR by x for routing towards a node z implies that x not only trusts y for routing (formula 2), but also trusts the choices of the routes made by y (formula 3). Actually, there is a chain of this indirect trust relation between x and any relay forwarding the packets to z. This chain has the particularity that only the last relay before z exchanges control messages directly with z (HELLO messages). This sequence expresses the recursiveness of MPR recommendations in OLSR, a property which allows us to deduce the following relation [26]:

$$x\ trusts.rec^{*}_{fw-z}\ (z)\ when.target[z]\ when.path[z] \tag{5}$$

This formula means that the routing target node is itself the starting point of the trust chain, and its *MPR* should be properly chosen so that all other nodes can correctly communicate with this node. This leads to a spreading of the trust placed in the MPRs.

Certain attacks against OLSR exploit the vulnerability resulting from the absence of validation of this derived trust chain. The node

should have a degree of mistrust concerning the information used for the calculation of the routing table. This mistrust could be associated to the validating procedure of the routing information which is spread in the network (TC messages).

5. Trust reasoning for securing OLSR protocol

In this section, we investigate how a node can detect misbehaving nodes by reasoning about information received from the network. Anomaly detection includes the consistency verification in OLSR messages (TC and HELLO) and trust-based reasoning that can be performed by each node in the network. Although it is a continuous process, the detection must progress from the reception of the link discovery messages to the construction of the routing table, giving the particular evolution of trust among nodes during these operations. In other words, a recursive and continuous checking of trust properties must be carried out, in order to validate all information received from the network, even after this node establishes a trust relation.

5.1. Validation of neighborhood discovery

The basic trust within OLSR is expressed by the fact that, according to the protocol specification [2], an OLSR node believes that a neighbor presents a correct identification (*ID*) and is committed to not interfering (*NI*) in the right protocol execution. Therefore, as pointed out by Wang et al. [24], this leads to intrinsic properties of the protocol regarding the expected correct behavior in message processing and routing organization. Concretely, a node x can believe that a neighbor y is generating correct messages on conformity to the OLSR specification if, by correlating messages coming from y, node x is able to verify the following properties:

1. MPR selectors of a node y (MSS_y) advertised in TC_y are symmetric neighbors of y.
2. When a node y advertises x in its TC message, being MPR of x, it has to be declared as MPR neighbor in $HELLO_x$.
3. When TC or DATA messages of a node y are forwarded, the forwarded message must be identical to the message generated by y.

The possibilities for an attacker to abuse these properties constitute the basic vulnerabilities of the protocol. However, using the concept of trust, we are able to reformulate these properties to express a mistrust reasoning that could be used by a node for self-protection against misbehaving neighbors. This leads us to a basic trust-based anomaly detection mechanism:

1. If y advertises it is MPR of other nodes, and it has not advertised them as symmetric neighbors on its HELLO message, then its behavior does not correspond to the OLSR specification. The neighbors (only the nodes that receive $HELLO_y$) have to mistrust y:

$$x \overset{HELLO_y}{\leftarrow} y, \quad x \overset{TC_y}{\leftarrow} y, \quad TC_y \not\subseteq NS_y \Rightarrow x \neg trusts(y) \tag{6}$$

2. If x detects that a distant node y advertising x on its TC message, but that it is not selected as MPR, then x must mistrust it, because its behavior does not correspond to the OLSR specification:

$$x \overset{TC_y}{\leftarrow} *, \quad x \in TC_y, \quad y \notin MPRS_x \Rightarrow x \neg trusts(y) \tag{7}$$

3. If x receives two TC messages from two different nodes with the same sequence number and originator address but with different information, then x must mistrust the two sender nodes:

$$\forall z, w \in MANET,\ x \overset{(TC_y)_z}{\leftarrow} *, \quad x \overset{(TC_y)_w}{\leftarrow} *, \quad (TC_y)_z \neq (TC_y)_w \Rightarrow x \neg trusts(\{z, w\}) \tag{8}$$

Indeed, even if a sender is the origin of the message TC_y, it may have an accomplice that sends another TC message to other nodes and which is different from the one that y sent to x. However, This rule can not be verified by nodes which receive only one TC message from the network. Especially nodes having the misbehaving node as the only neighbor, they will trust only the messages coming from it, and will have a wrong view of the network topology.

Note that the mistrust reasoning cannot every time allow the precise identification of the misbehaving node, but allows the detection of a behavior anomaly related to a group of nodes which includes the attacker, as indicated in formula 8. Also, it is important to mention that it is possible to receive consistent *HELLO* and *TC* messages from an attacker advertising wrong information. This situation can be detected when the node compares and correlates messages received from different nodes (see next section).

These properties can also be extended to allow nodes to check the local vision of their neighbors about the network. For example, if x detects a remote node y advertising one of its neighbors z in the message TC_y, but if z did not declare y as symmetric neighbor, then it must mistrust y because its behavior does not correspond to the OLSR specification.

Similarly, the neighborhood discovery (links) must be consistent among all nodes. One of the problems that can occur in this operation is the reception of contradictory HELLO messages from different neighbors. This may be a temporary situation where a group of nodes is at the beginning of a discovery process. However, if the situation arises after an initialization process, the node which continues to receive contradictory messages should mistrust the nodes which generate these messages. The neighborhood discovery can be validated by checking links advertised in received HELLO messages. Given the fact that if a symmetric link between two nodes is advertised by one node, it must also be advertised as such by the other. In terms of trust, this means a node cannot be sure of the existence of links between its neighbors if it receives conflicting information on this link.

Thus, it is possible to establish basic controls for the detection of neighbors with abnormal behavior that can increase the robustness of the protocol and allow the detection of possible direct attacks. At this stage, these checks are not enough to counter other attacks and do not allow a global cooperation for the detection of abnormal behavior.

5.2. Validation of MPR selection

MPR selection is a critical operation, it provides the access to the network routing infra-structure for each node and allows them to be known by other distant nodes. In the specification of OLSR [2], there is no verification of MPR behavior. This vulnerability is exploited by many attacks through which the attacker aims to be selected as MPR by a target node in order to control the target node input–output message flows [33].

The danger with MPR selection based on the degree of reachability comes from the fact that any attacker can give wrong information about its neighbors which cannot be verified. Thus, to strengthen the MPR selection, we introduce trust-based reasoning to counter attacks. In our approach, after the MPR selection each node should verify the two following points:

1. The correct behavior of an MPR (as specified by OLSR).
2. The local choices of MPRs by a node must be in accordance to global topology information received by this node.

Q1 Please cite this article in press as: A. Adnane et al., Trust-based security for the OLSR routing protocol, Comput. Commun. (2013), http://dx.doi.org/10.1016/j.comcom.2013.04.003

We propose for each node to use the concept of trust to supervise the behavior of its selected MPR. According to the OLSR specification, the correct behavior of an MPR, regarding routing, is defined by two operations: TC message generation and Forwarding DATA packets and TC messages of the MPR selectors. If a node is able to validate these functions by observing the MPR behavior, then the node can consider its trust relation with this MPR correct, as expressed in (2).

Conversely, if these functions cannot be validated for a selected MPR node, the trust relation is broken and the selection of this node as MPR must be canceled:

Checking TC message generation:
If a selected MPR (y) does not generate TC messages correctly advertising its MPR selectors, each selector (x) must mistrust this MPR:

$$y \in MPRS_x, \quad (x \overset{TC_y}{\nleftarrow} y) \quad or \quad (x \overset{TC_y}{\leftarrow} y, \quad x \notin TC_y) \Rightarrow x \neg trusts(y) \tag{9}$$

Checking data packet and TC message forwarding:
If a selected MPR (y) does not forward data packets ($DATA_x$) and TC messages (TC_x) sent by its MPR selectors, each selector (x) must mistrust this MPR:

$$y \in MPRS_x, (x \overset{TC_x}{\rightarrow} *, x \overset{(TC_x)_y}{\nleftarrow} y) or (x \overset{DATA_x}{\rightarrow} *, x \overset{(DATA_x)_y}{\nleftarrow} y) \Rightarrow x \neg trusts(y) \tag{10}$$

This validation can be used to detect the attacks aimed at falsely modifying the topology by means of TC message fabrication or modification. Also, this can be used against the black-hole attack, where an attacker MPR drops data messages instead of forwarding them.

Following the MPR selection, the nodes in $MPRS_x$ are required to recommend x the routes to the distant nodes z (presented in formula 3). The formula presents the general rule of derived trust for the routing in the networks operating under OLSR. Indirectly, the nodes establish a trust relation with a distant node (n hops away, more than 2 hops) by chained recommendations.

In order to allow neighbor behavior verification by each node, we have to check if these neighbors are correctly performing their MPR selection and routing table calculation. Moreover, each node has to verify the consistency of advertised information with its local network vision.

According to the MPR selection algorithm specified in OLSR [2], if two neighbors, x and y, have the same neighbors (NS), they should also select the same MPRs (except in the situation where the different MPRs have the same neighbors). This property allows x and y (and actually any common neighbors of x and y – $NS_x \cap NS_y$) to compare MPR selection between x and y. This comparison is related to an interesting situation which generally arises during the attacks against MPR selection. In this situation, an attacker A wants to control the traffic of a target x and so as to be selected as the only MPR, the attacker advertises all two-hop neighbors of the target adding a fictitious node as its own direct neighbors ($NS_A = NS_x \cup 2HNS_A \cup z$). Generally, when an attacker A wants to control the network traffic flowing from a target x to another node B, this attacker will advertise as neighbors those MPRs selected by the target to reach node B. This situation is expressed as follows:

$$x \in NS_y, \quad y \in NS_x, \quad NS_x - \{y\} \subseteq NS_y - \{x\} \Rightarrow MPRS_x - \{y\} \subseteq MPRS_y - \{x\} \quad or \quad \forall z \in MPRSx, \exists w \in MPRS_y : z \neq w, \quad NS_z = NS_w \tag{11}$$

Reachability is another criterion for MPR selection: the algorithm specifies each node must begin by selecting as MPR the neighbors which are the only nodes to provide reachability to certain nodes in the two-hop neighbor set. Since the symmetric links of neighbors are not verified, a malicious node can advertise an additional unused address (z) with symmetric link status. Hence, on the reception of a malicious node's HELLO message, all its neighbors will choose it as MPR without any existing proof of that link. It is difficult to verify the links status of all the neighbors. Nevertheless, given the important role of MPR nodes as routers in the network, this verification is essential. In OLSR, we can say that trust level is more important on the MPR nodes than on the other neighbors [26], thus the MPRs present a higher risk if they are compromised. Consequently, this reinforces the need to continuously check the information provided by the MPR nodes.

There are other properties that can be checked on the reception of TC messages from a neighbor. These properties allow a node to validate its own MPR selection and to verify the MPR selection made by its neighbors:

- Two nodes x and y with the same neighborhood (the same symmetric neighbors) cannot be both selected as MPRs by common neighbors. Because, if one is selected, for example x, it will provide reachability to its neighborhood (NS_x) which covers y's neighborhood (NS_y).
- If the neighborhood of a node x includes in the neighborhood of another node $y : NS_y \subset NS_x$, then y should not be selected as MPR, all its neighbors must select x as MPR (or another neighbor with larger neighborhood).

Consequently, when a node x detects an inconsistency related to the above properties, it has to mistrust the nodes that generate this information:

$$x \overset{HELLO_A}{\leftarrow} A, \quad x \overset{HELLO_B}{\leftarrow} B, \quad \exists z \in MSS_A \cap MSS_B, \quad NS_A \subseteq NS_B \Rightarrow x \neg trusts(\{A, B, z\}) \tag{12}$$

Note that x has to mistrust A, B and z. x cannot precisely detect the misbehaving node, because z may not select its MPRs correctly in order to use both nodes A and B as router, or it may be that A or B claim to be MPR of z though they are not.

Once again, it is worth pointing out that the mistrust reasoning does not always allow the precise identification of the misbehaving node. However, it allows the detection a behavior anomaly related to a group of nodes which includes the attacker.

5.3. Synthesis

Correlation between received messages allows the OLSR nodes to validate their local vision and that of their neighbors. When the received information is coherent with OLSR specification, this reasoning enables the creation and validation of trust relations. Otherwise, this reasoning allows to mistrust the sources of these inconsistencies (malicious nodes).

In our previous works, we assumed that the identity of each node cannot be usurped, yet in OLSR, the identity is equivalent to the IP address and can be easily spoofed. To resolve this problem and ensure the assumption, we propose a solution to prevent identity usurpation, based on provable identity.

This reasoning only represents detection steps, and some nodes are not able to detect the source of the inconsistency. In addition, no measures are taken to solve the case of an inconsistency and counter attack. It is therefore important to investigate possible countermeasures that a node can apply to solve the problems detected. Nevertheless, the prevention based on provable identity is not enough to resolve all inconsistencies. For this, we offer other general countermeasures to isolate the attacker if the attack succeeded despite preventive measures.

Q1 Please cite this article in press as: A. Adnane et al., Trust-based security for the OLSR routing protocol, Comput. Commun. (2013), http://dx.doi.org/10.1016/j.comcom.2013.04.003

6. Provable identity for neighborhood validation

In OLSR, identity usurpation is easy to perform, and it is difficult to verify the correctness and the existence of the advertised links. In *SOLSR* [20], the authors assume that nodes are either trusted or mistrusted, and that trusted nodes are not compromised. Their approach is based on the signature and time-stamp of each OLSR control message (*SOLSR* is presented in *Related works* section). The signature is sent in the same packet with the associated message.

Each trusted node in the network would be able to acquire the keys required for the verification of signatures of any other node. To do this, the authors propose two different Public Key Infrastructures (*PKI*): proactive and reactive. Both proposals use a certification authority that allows to define trusted nodes and distribute their public keys. In the case of using a public-key system, the problem is thus the key distribution/certification in a way such that the public keys can be trusted. In our view, the principle of ad hoc networks is not compatible with using certification authority. Therefore, we propose to use provable identity mechanism with *SOLSR* to avoid public keys certification [30].

6.1. Provable identity for OLSR with IPv4

A provable identity is an identity which is easy to verify, but difficult to usurp. It is not based on a certification authority to prove the identity of each node. The objective is not to authenticate the other nodes, but once the identity of each node is detected, it ensures that the following messages, provided by the same entity, will be authenticated, and that this identity cannot be usurped by another node [34,35].

Usually, the public key is used as a provable identity: a node being identified with its public key can sign a challenge using its private key, and it is the only node able to decrypt a message encrypted with that public key. Provable identity is an authentication mechanism adapted to the decentralized nature of ad hoc networks, because it is based on asymmetric cryptography and does not involve a certification authority. It allows each node in the ad hoc network to be identified and authenticated by other entities of the network.

OLSR protocol use the IP address to identify nodes and assumes that each node uses a unique address. However, the IP address does not guarantee the identity of each node as it can be easily spoofed, hence the need for a link with the identity encryption (public and private key). In IETF (Internet Engineering Task Force), some works have suggested to integrate the mechanism of provable identity in IPv6, in order to bind the interface identifier with the public key of the node. In short, the solution consists in applying a hash function on a public key to obtain its IP address and, at the same time, its provable identity. Therefore, if OLSR is applied above IPv6, then provable identity can be obtained simply by integrating the extension proposed by the IETF [36]. Otherwise (from OLSR on IPv4), we present in the following paragraph an approach to have a provable identity for OLSR on IPv4, and this without changing the OLSR message structure and routing table. The provable identity mechanism ensures that an IP address is unique and cannot be misused. Our approach is based on message signature (*SOLSR*). We require that the message signature and the corresponding public key are attached to the same packet. The signature and public key of the node are added to the OLSR packet as a particular type of message (Fig. 4).

When exchanges begin, each node A sends its public key K_{Pub_A} to other entities that will store the hash of this public key as the provable identity IdP_A of A. Before sending packets, each node A signs its messages using its private key K_{Pri_A}, and when these messages are received, it is possible to verify their integrity and authenticity by checking their signatures with the public key of the sender. After the verification of message integrity, the provable identity of the sender will take the following format (H: Hash function): $IdP_A = (H(K_{Pub_A}))$,

Consequently, if two nodes use the same IP address, they are considered as two different identities because they do not have the same public key. So, they have to change their IP address to be accepted into the network. To avoid this situation and ensure that an IP address is used only by one node, we propose the use of a table *TabIP* index on IP addresses which associates each public key to one IP address: $K_{Pub_A} = TabIP(@IP_A)$ (see Table 1). Q2

Therefore, if a node x with the public key K_{Pub_x} has the same IP address as another node A, the $TabIP_n$ table returns the public key of A ($TabIP(@IP_A) = K_{Pub_A}$) which was previously saved and which is different from the public key of x ($K_{Pub_x} \neq K_{Pub_A}$). Then messages of x will be rejected by n and x has to change its IP address to integrate the network. If a node detects that another node uses the same IP address, then it has to change it.

On the other hand, if a node changes its IP address without changing the public key, it will be detected as another entity, because no public key has been registered for the new IP address. This means that verification of identity in the table IP address is not sufficient and it is important to check if its provable identity (hash of its public key) has already been detected. To do this, we propose the use of a second table, called hash table *TabH* (Table 2), which is indexed on the hash of public key. This represents an optimization because it allows reasoning on the hash and not on the public key: $@IP_A = TabH(H(K_{Pub_A}))$.

In this case, if a node changes its IP address without changing its public key, then the new IP address will be updated without changing the provable identity.

In brief, the verifications of IP address in $TabIP_n$ table and public key in $TabH_n$ table ensure to the node n that one IP address can be used only by a single node, and if a node changes its IP address without changing its public key it will be detected having the same provable identity. The use of the two tables $TabH_n$ and $TabIP_n$ allows us to represent the provable identity as the hash of the public key:

$$K_{Pub_A} = TabIP_n(@IP_A) \quad \text{and} \quad @IP_A = TabH_n(H(K_{Pub_A})) \Longleftrightarrow IdP_A = H(K_{Pub_A}) \tag{13}$$

6.2. Proof of neighborhood

Relying on message signature (the solution proposed by *SOLSR*) and provable identity, we propose a preventive method to secure the neighborhood discovery. This method ensures the integrity of the information provided by symmetric neighbors, and thus validates the neighbor discovery (one-hop and two-hop neighbors) and MPR selection.

In the following, we will reconsider the previous formulas with the provable identity notation, for example: $B \in NS_A$, signifies that $@IP_B \in NS_A$ but $TabIP_A(@IP_B) = K_{Pub_B}$ and $TabH_A(H(K_{Pub_B})) = @IP_B$,

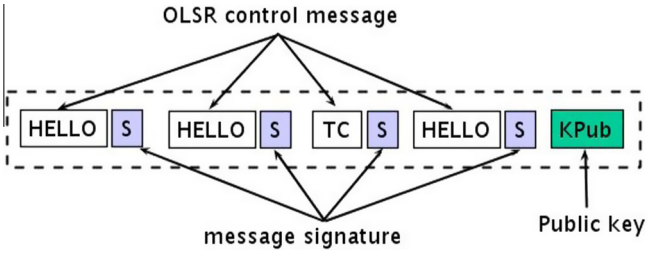


Fig. 4. Packet format in *SOLSR*.

Table 1
Structure of the IP address table $Tab\ IP_n$ of the node n.

@IP	Public key
A	K_{Pub_A}
B	K_{Pub_B}
...	...

Table 2
Structure of the hash table $Tab\ H_n$ of the node n.

Hash of public key	@IP
$\mathbf{H(K_{Pub_A})}$	A
$\mathbf{H(K_{Pub_B})}$	B
...	...

and A *trusts* (B) signifies that A trusts the node which has the provable identity $IdP_B = H(K_{Pub_B})$.

After packet reception, each node detects the identity of the sender. For example, node B detects the identity of its neighbor A from the packet $P_A = HELLO_A + signHello_A + K_{pub_A}$ as follows:

$$If\ P_A\ is\ valid\ \Rightarrow \begin{array}{c} TabIP_B(@IP_A) = K_{Pub_A} \\ TabH_B(H(K_{Pub_A})) = @IP_A, IdP_A = H(K_{Pub_A}) \end{array}$$

To prove to its neighbors (including A) that it is the symmetric neighbor of A, we propose for B to send a message $Proof_B$including a $HELLO$ message signed by A where $(B, sym) \in LS_A$.

$$Proof_B \leftarrow Proof_B \cup \{HELLO_A + signHello_A + K_{Pub_A}\}.$$

Thus, any neighbor of B (including A) will verify that A is effectively neighbor of B and will declare A as 2-hop neighbor. Besides, A can verify its own link with B and any links of B with its other neighbors. If A finds that B has the proof of the reception of its HELLO message $HELLO_A$, it will establish a symmetric link with B and advertise it.

After establishing a new symmetrical link, the proof is sent only once,[1] to prove the validity of the new symmetrical link to other neighbors and to prove the validity of the other symmetrical links to the new neighbor.

After receiving $Proof_B$ from B (the structure of the packet is presented in Fig. 5), all the symmetric neighbors of B will proceed to the following verification $(\forall z \in NS_B)$:

- Check the integrity of the message $Proof_B$: node z must first verify the integrity of each message. For example, for the $HELLO_A$ message, it checks the consistency of the hash with the message signature $signHello_A$ using K_{Pub_A}. If the message is validated then z goes to the next step. Otherwise, the message is rejected and it is impossible to prove that B is really neighbor of A, then node A will not be considered as 2-hop neighbor.
- Verification of symmetrical link: node z checks the symmetrical link to ensure that node A also has declared B as symmetrical neighbor:
  1. If B is declared as symmetrical neighbor in the $HELLO$ message of A $(HELLO_A \in Proof_B$ and $(B \in NS_A$ or $B \in MPRS_A))$, then the link between A and B is validated, and node A is inserted in the 2-hop neighbor set of z.
  2. Otherwise, there is an inconsistency and the link is not validated, and node B is reported as malicious node, establishing the following relationship of mistrust:

[1] The message $Proof_B$ is sent only by node B, and must not be broadcasted. This message will not be distributed throughout the network.

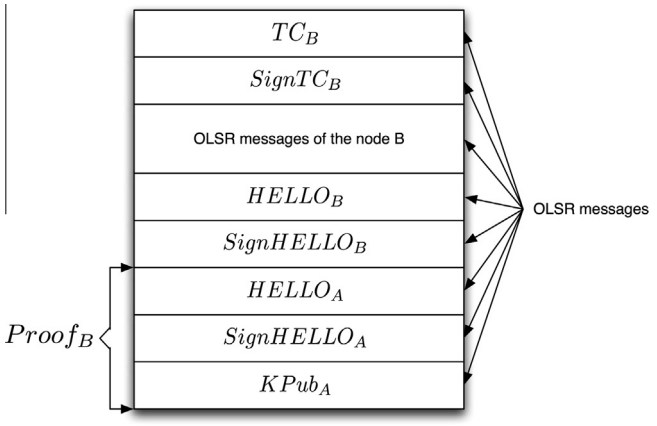


Fig. 5. Structure of OLSR packet provided by node B and including its proof of neighborhood with the node A.

$$z \overset{HELLO_B}{\leftarrow} B, [A \in NS_B \cup MPRS_B], \quad z \overset{Proof_B}{\leftarrow} B, [B \notin NS_A \cup MPRS_A] \Rightarrow z\neg trusts(B) \tag{14}$$

With this mechanism, the reception of the $HELLO$ message is not sufficient to detect the 2-hop neighbors. For any link between symmetrical nodes x and y, each node z neighbor of x must receive a valid proof $Proof_x$ including $HELLO_y$ and claiming x also as symmetrical neighbor. Only nodes that we have the proof of neighborhood are inserted, in order to obtain *2-hop trusted neighborhood*. Consequently, attacks against the 2-hop neighborhood become inefficient because a node cannot lie about its neighbors (nodes in the trusted neighborhood). This preventive mechanism will cover the rules of trust on the local vision and will enable a quicker detection on link consistency.

7. Countermeasures

The first countermeasure concerns basic operations in OLSR (neighborhood discovery and MPR selection) while the second countermeasure concerns the distribution of information about trust relations and attack detection to alert the other nodes.

In both solutions, we suppose that the time-stamp mechanism proposed by *SOLSR* [20] and the provable identity mechanism presented previously are set up respectively to ensure the freshness and authentication of messages.

In OLSR, when a node detects a symmetric neighbor, it considers the information provided as correct and establishes a trust relation. Using trust-based reasoning, when a symmetrical neighbor is detected as malicious and does not respect the OLSR specification, the trust relation presented in formula (1) is broken. As a countermeasure, we propose to reject messages of this neighbor and break its symmetrical link. Thus, it will be isolated and the information it sends will be ignored:

$$x\neg trusts(y) \Rightarrow MN_x = MN_x \cup \{y\}, \quad reject\ HELLO_y\ and\ TC_y \tag{15}$$

Where MN_x (Mistrusted Nodes) is the set of malicious nodes detected by x. When an inconsistency is detected, the provable identity of the attacker node is inserted in this set. Thus, even if the attacker node IP address changes, it will always be isolated. This countermeasure is taken only when the detection is exact, and it may be permanent or temporary depending on the user needs and the context of use. Temporary countermeasures isolate the mistrusted node for a certain period of time, and allow the mistrusted node to gain the trust of the network if he behaves correctly. When the mistrust is partial, no rule is applied (partial mistrust help nodes

to detect general misbehavior related to group of nodes, no countermeasures is taken until exact mistrust is reached).

When an MPR node is detected as malicious, the first countermeasure to take is the same countermeasure as that taken when a symmetric neighbor is malicious (15) in order to break the symmetric link. Secondly, it must update its MPR selection (to delete the malicious node from the MPR set). It is important to point out that there are certain cases where the malicious neighbor is the only MPR to provide access to certain nodes of the network, and breaking symmetrical link with this neighbor means that the node cannot access the network accessible through the malicious neighbor.

The second countermeasure we propose is to share information about detected attacks by broadcasting an alert to notify all nodes in the network to mistrust the attacker. However, to avoid false alerts, the alert must provide reliable proofs of the attack that cannot be falsified. When the detection is exact, we propose to broadcast messages that have revealed inconsistencies and allow detection of the attack. Moreover, provable identity and messages signing (*SOLSR*) ensure the authenticity and integrity of each message and ensure the reliability and integrity of alerts. It is important to note that the alert is not a new type of message, but that it represents the broadcast of control messages which revealed an inconsistency.

Trust reasoning, for inconsistency detection, is based on the comparison of provided information (*HELLO*, *TC* and DATA messages) with local vision. Therefore, the alert consists in broadcasting the two messages that have revealed the inconsistency (including *HELLO* messages) and allow the attack detection according to the two following scenarios:

1. When a node detects inconsistency between received message and local vision, it must notify the other nodes by forwarding the received message, and the message representing its local vision (*HELLO* or *TC*). For example, if a node detects inconsistency between its MPR selection and the received message, then it must forward the received message and its *HELLO* message, because the *HELLO* message represents the MPR selection. Also, if the inconsistency concerns MPR selectors, then it must retransmit its *TC* message instead of the *HELLO* message.
2. When a node detects inconsistency between two messages provided by other nodes, it has to alert the network by retransmitting these two messages. For example: if a node detects that its neighbor *x* advertises certain nodes in its *TC* message, but does not advertise them in its *HELLO* message, then it detects inconsistency between the *HELLO* and *TC* messages of *x*, and has to broadcast these messages.

It is important to note that the first countermeasure (as defined in the previous section) must be set up before sending alert messages. Precisely when the attacker is MPR, we must break the symmetrical link and update the MPR selection to ensure that the alert reaches all the nodes of the network. On the other hand, even if the *TC* messages are expected to reach all nodes of the network, when the attacker is MPR, it can disrupt routing, and therefore there is no guarantee that these messages reach all nodes. So when the alert includes *TC* messages and concerns MPR node, it is important to retransmit these messages after the detection of the attack and the change in the MPR selection.

Each node receiving the alert will apply the same reasoning (detect a malicious node). It will compare the information provided in the alert with the local vision before the broadcast. Furthermore, before broadcasting alerts, the provided information must be correlated with the local vision to detect false alerts and refuse to broadcast alerts generated by malicious nodes. For example if *y* has already detected *x* as malicious node, then it should not broadcast the alert generated by *x*. Note that the distribution of a trust proof allows nodes that have not detected the attack to detect it, and also allows exact detection for nodes that have established a partial detection of the attack.

These countermeasures imply the retransmission of *HELLO* messages when an alert is broadcasted, and consequently, on receiving these messages an asymmetric link is created with the *HELLO* message source. This indirect consequence does not affect the routing operations, since asymmetric links are not considered for the MPR selection and the routing table calculation, and are deleted after a certain time if the symmetrical link is not created.

Furthermore, these countermeasures involve the back up of received messages during a specific period in order to compare them with messages received thereafter. The backup process is applied to each received message, and only saves the important information (e.g. the message type (*HELLO*, *TC* or DATA), the message freshness (according to the time-stamp in *SOLSR*), the sequence number, the source address, and the list of nodes declared in the message and the type of link for *HELLO* messages). If the message is already stored with the same sequence number or an inferior number, then its contents and validity time are updated. If the message is received for the first time then it is stored with a validity time. When the validity time has expired the message is deleted.

8. Simulation results of trust-based reasoning

We have used the *GlomoSim Simulator* and the OLSR patch developed by the *Niigata University* to simulate the attacks and previous formulas. We have added to this patch a module implementing trust rules, and several attack scenarios. In our simulations, ad hoc networks are composed of 50 nodes which are placed randomly. Moreover, the attackers are selected randomly, and each one selects an attack scenario, as well as a set of targets according the selected attack. However, since the ad hoc networks have to be stabilized to allow the attacker to perform the attacks, we have considered that nodes are not mobile. The following attacks was implemented: generation and modification of HELLO messages, generation and modification of TC messages, black-hole attack and identity spoofing attack (see Table 3).

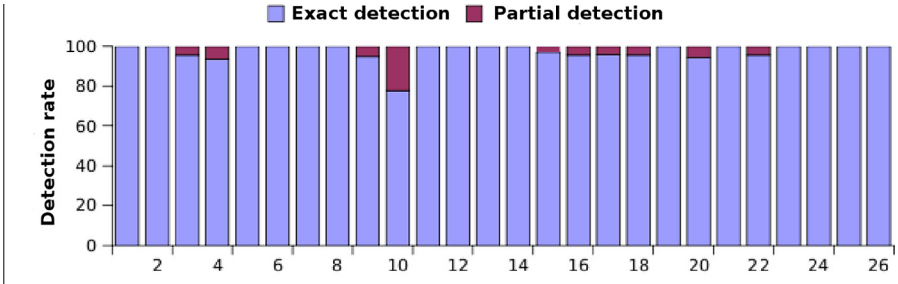
In the following, we discuss only results with 50 nodes using the first attack scenario which takes place according to the following steps:

1. The attacker *A* identifies target *T*, its neighbors and 2 hop neighbors.
2. The attacker *A* detects its common neighbors with the target *T*, and modifies its *HELLO* messages to advertise their neighbors as symmetric neighbors as well as an additional fictitious node *X*.
3. The attacker advertises the target's 2 hop neighbors in its TC messages.

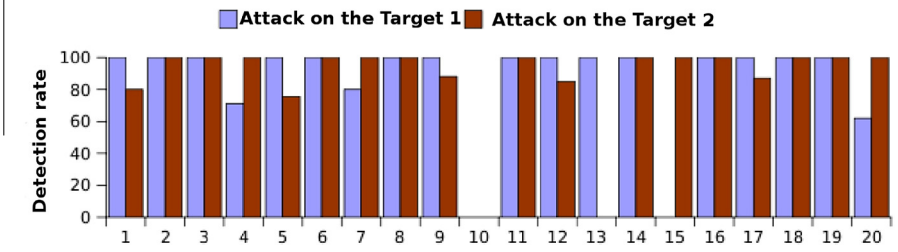
According to OLSR specification, the target has to select the attacker as MPR because it provides reachability for the node *X*, and so the attacker can control some target's flows.

Simulation results are presented in the figures Table 3A–C. In the case of one attacker and one target (Table 3A), the detection rate is usually 100%. However, there is one case where the detection rate was 0%, this is the case where the target is completely isolated from other network nodes and/or when the attacker is always a unique MPR of the target (even before applying the attack). It is obvious that in this situation the objective of such an attack makes no sense because the attacker is already MPR unique of the target and can therefore control all its messages. The realization of the attack does not change the network topology, and therefore it is not

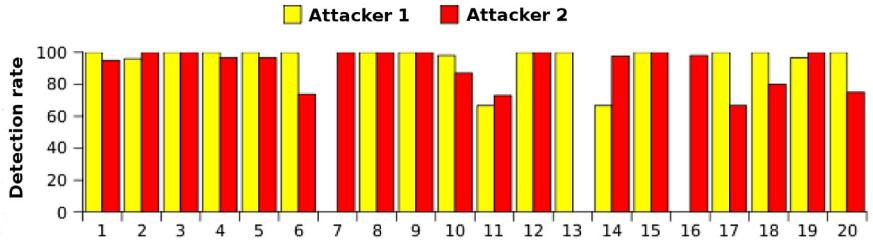
Table 3
Simulation results.



A- Detection rate for 26 simulations : 50 nodes, 1 attacker and 1 target



B- Detection rate for 20 simulations : 50 nodes, 1 attacker and 2 targets



C- Detection rate for 20 simulations : 50 nodes, 2 attackers and 1 target

detected. The only point on which the attacker has lied, is its link with the fictitious node (which has no relevance in this situation). This lie can not be detected because a node can be the unique MPR node of other nodes. Thus, a malicious node can create as many fictional neighbors it wants, since it will be their only symmetric neighbor, and the other nodes of the network have no way to verify the physical existence of the latter.

This attack represents a passive listening, because it allows the attacker to analyze messages flow of the target, and it will be detected when the attacker is active. For example, by refusing to retransmit packets.

In the case of multiple targets (Table 3B) and attackers (Table 3C), the results of detections relate the attack against each target separately. attackers are completely independent. It is important to note that in some simulations the detection rate is lower than 100% for several reasons:

- In some scenarios, the simulation time is not enough for all nodes to establish a global vision of the network, and so they are not able to detect the attack.
- The simulator chooses the attacker randomly. Thus, the establishment of the attack may fail if the attacker has no neighbor, or if the attacker is the unique MPR of the target.

Moreover, some nodes are unable to precisely detect the attacker (partial detection). This situation can be resolved by a system of distribution of trust proof (countermeasures, Section 7). Thus, nodes that detect the attack exactly distribute the attack proof and enable other nodes to also detect the malicious nodes.

9. Conclusion

We have presented a trust-based solution for securing the OLSR Ad hoc routing protocol in three steps. The first step was the analysis of the implicit trust relations in OLSR. This analysis highlights the possible measures to make OLSR more reliable by exploiting the operations and information already existing in the protocol.

To detect misbehaving nodes, we have developed in the second step, trust-based reasoning by correlating information provided in the OLSR messages received from the network. The integration of this reasoning allows each node to check the consistency of the behavior of other nodes and validate trust relationships established implicitly.

Finally, the third step complements the second by offering two complementary solutions: prevention to resolve certain vulnerabilities of OLSR protocol, and countermeasures to stop and isolate malicious nodes. These proposals correspond to the trust reasoning that has been done by each node. Simulation results illustrate the effectiveness of trust-based reasoning and countermeasures to stop and isolate misbehaving nodes.

In summary, we demonstrated that our solution allows to verify if the behavior of other nodes in the network complies with the

Q1 Please cite this article in press as: A. Adnane et al., Trust-based security for the OLSR routing protocol, Comput. Commun. (2013), http://dx.doi.org/10.1016/j.comcom.2013.04.003

specification of OLSR. This result allowed us to ensure efficient routing operations and ensure the validity of the network topology (routing table). Our approach brings few modifications on OLSR packets, and it is still compatible with the bare OLSR and SOLSR.

These results motivate extending the approach for establishing more example of simulation and to check degree of automation of reasoning about trust in order to measure the impact of these solutions on Qos of the protocol, while preserving the self-organization and the dynamic environment of ad hoc network. Moreover, our approach can be applied more generally to other protocols in spontaneous and self-organized environment. The explicit specification of trust relations leads to identify whether the underlying assumptions for the operation of a protocol are realistic or not. Moreover, explicit trust-relations can be used in formal analysis of the correction of protocol vulnerabilities.

References

[1] M. Abolhasan, T. Wysocki, E. Dutkiewicz, A review of routing protocols for mobile ad hoc networks, Ad Hoc Networks, 1570-8705 2 (1) (2004) 1–22.
[2] T. Clausen, P. Jacquet, IETF RFC-3626: Optimized Link State Routing Protocol OLSR, 2003.
[3] S. Marsh, Formalising Trust as a Computational Concept, Ph.D. Thesis, Department of Mathematics and Computer Science, University of Stirling, 1994.
[4] S. Marti, T.J. Giuli, K. Lai, M. Baker, Mitigating routing misbehavior in mobile ad hoc networks, Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking (MobiCom '00), ACM, New York, NY, USA, 2000, pp. 255–265.
[5] S. Choudhury, S. Deb Roy, S.A. Singh, Trust Management in Ad Hoc Network for Secure DSR Routing, Novel Algorithms and Techniques in Telecommunications, Automation and Industrial Electronics, Springer, 2008. pp. 496–500.
[6] S. Buchegger, J-Y. Le Boudec, Performance analysis of the confidant protocol: cooperation of nodes – fairness. Dynamic Ad-hoc networks, Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC), IEEE, 2002.
[7] P. Michiardi, R. Molva, Core: a collaborative reputation mechanism to enforce node cooperation in mobile ad hoc networks, in: IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security: Advanced Communications and Multimedia Security, 2002, pp. 107–121.
[8] A. Boukerche, Y. Ren, A trust-based security system for ubiquitous and pervasive computing environments, Elsevier – Computer Communications, Secure Multi-Mode Systems and their Applications for Pervasive Computing, 0140-3664 31 (18) (2008) 4343–4351.
[9] A.A. Pirzada, A. Datta, C. McDonald, Incorporating trust and reputation in the DSR protocol for dependable routing, Elsevier – Computer Communications, 0140-3664 29 (15) (2006) 2806–2821.
[10] K. Meka, M. Virendra, S. Upadhyaya,Trust based routing decisions in mobile ad-hoc networks, in: Workshop on Secure Knowledge Management (SKM), 2006.
[11] S. Buchegger, J-Y. Le Boudec, The effect of rumor spreading in reputation systems in mobile ad-hoc networks, in: Wiopt-03: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, Sophia-Antipolis, France, 2003.
[12] L. Buttyan, J-P. Hubaux, Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks, in: Technical Report DSC/2001/001, Swiss Federal Institute of Technology – Lausanne, Department of Communication Systems, 2001.
[13] M.-Y. Su, WARP: a wormhole-avoidance routing protocol by anomaly detection in mobile ad hoc networks, Elsevier – Computers & Security, 0167-4048 29 (2) (2010) 208–224.
[14] P. Narula, S.K. Dhurandher, S. Misra, I. Woungang, Security in mobile ad-hoc networks using soft encryption and trust-based multi-path routing, Elsevier – Computer Communications. Algorithmic and Theoretical Aspects of Wireless ad hoc and Sensor Networks, 0140-3664 31 (4) (2008) 760–769.
[15] S. Lee, B. Han, M. Shin, Robust routing in wireless ad hoc networks, in: International Conference on Parallel Processing Workshops (ICPPW G02) 2002, IEEE Computer Society, 2002, pp. 18–21.
[16] M G. Zapata, N. Asokan, Securing ad hoc routing protocols, in: Proceedings of the 2002 ACM Workshop on Wireless Security (WiSe), 2002, pp. 1–10.
[17] Y. Hu, A. Perrig, D. Johnson, Ariadne: a secure on-demand routing protocol for ad hoc networks, Wireless Networks, vol. 11, Kluwer Academic Publishers, 2002. pp. 21–38.
[18] M. Omar, Y. Challal, A. Bouabdallah, Reliable and fully distributed trust model for mobile ad hoc networks, Elsevier – Computers & Security, 0167-4048 28 (3–4) (2009) 199–214.
[19] T. Wan, E. Kranakis, P.C. Van Oorschot, Securing the destination-sequenced distance vector routing protocol (S-DSDV), in: Sixth International Conference on Information and Communications Security (ICICS), LNCS, vol. 3269, Springer, 2004, pp. 358–374.
[20] C. Adjih, T. Clausen, P. Jacquet, P. Mhlethaler, A. Laouiti, D. Raffo, Securing the OLSR protocol, in: Second IFIP Annual Mediterranean Ad Hoc Networking Workshop (Med-Hoc-Net), 2003, pp. 25–27.
[21] D. Raffo, C. Adjih, T. Clausen, P. Mhlethaler, An advanced signature system for OLSR, in: Proceedings of the 2hn ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN), 2004, pp. 10–16.
[22] R. Puttini, J-M. Percher, L. Mé, R. Jr. De sousa R, A fully distributed IDS for MANET, ISCC 2004, Ninth International Symposium, vol. 1, IEEE Computer Society, 2004, pp. 331–338.
[23] C.V. Zhou, C. Leckie, S. Karunasekera, A survey of coordinated attacks and collaborative intrusion detection, Elsevier – Computers & Security, 0167-4048 29 (1) (2010) 124–140.
[24] M. Wang, L. Lamont, P. Mason, M. Gorlatova, An effective intrusion detection approach for OLSR MANET protocol, in: Proceedings of the First International Conference on Secure Network Protocols (NPSEC), IEEE Computer Society, Washington, USA, 2005, pp. 55–60.
[25] N. Cuppens-Boulahia, S. Nuon, F. Cuppens, T. Ramard, Property based intrusion detection to secure OLSR, in: Proceedings of the Third International Conference on Wireless and Mobile Communications, IEEE Computer Society, 2007, p. 52.
[26] A. Adnane, R. De Sousa, C. Bidan, L. Mé, Analysis of the implicit trust within the OLSR protocol, in: IFIPTM07: Joint iTrust and PST Conferences on Privacy, Trust Management and Security, IFIP International Federation for Information Processing, vol. 238, Springer, 2007, pp. 75–90.
[27] A. Adnane, R. De Sousa, C. Bidan, L. Mé, Autonomic trust reasoning enables misbehavior detection in OLSR, in: The 23rd Annual ACM Symposium on Applied Computing (ACM SAC), vol. 3, 2008, pp. 2006–2013.
[28] A. Adnane, C. Bidan, R. De Sousa, Validation of the OLSR routing table based on trust reasoning, in: International Workshop on Trust in Mobile Environments (TIME 08) co-located with iTrust/PST, 2008, pp. 1–12.
[29] A. Adnane, C. Bidan, R. De Sousa, Effectiveness of trust reasoning for attack detection in OLSR, in: International Workshop on Security in Information Systems (WOSIS), INSTICC Press, 2008, pp. 151–157.
[30] A. Adnane, C. Bidan, R. De Sousa, Trust-based countermeasures for securing OLSR protocol, in: IEEE/IFIP International Symposium on Trusted Computing and Communications (TRUSTCOM2009), vol. 2, Canada, 2009, pp. 745–752.
[31] R. Yahalom, B. Klein, T. Beth, Trust relationships in secure systems – a distributed authentication perspective, in: SP'93: Proceedings of the 1993 IEEE Symposium on Security and Privacy, IEEE Computer Society, Washington, USA, 1993, pp. 150–164.
[32] T. Grandison, M. Sloman, A survey of trust in internet applications, IEEE Communications Surveys and Tutorials 3 (4) (2000) 2–16. 4th Quarter.
[33] Y-C. Hu, A. Perrig, A survey of secure wireless ad hoc routing, IEEE Security and Privacy 2 (3) (2004) 28–39.
[34] C. Montenegro, C. Castelluccia, Statistically unique and cryptographically verifiable (sucv) identifiers and addresses, in: NDSS02, 2002.
[35] G. O'Shea, M. Roe, Child-proof authentication for mipv6 (cam), in: ACM SIGCOMM Computer Communication Review, 2001, pp. 4–8.
[36] A. Laouiti, S. Boudjit, P. Minet, C. Adjih, Olsr for ipv6 networks, Med-Hoc-Net, June 2004.