UNIVERSITÀ DELLA CALABRIA

UNIVERSITA' DELLA CALABRIA

Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica

Dottorato di Ricerca in

Information and Communication Engineering for Pervasive Intelligent Environments

CICLO

XXIX

TRUST MANAGEMENT ANALYSIS AND PROPOSAL OF TRUST-BASED ENERGY-EFFICIENT INTRUSION DETECTION SYSTEM FOR WIRELESS AD-HOC NETWORKS

Settore Scientifico Disciplinare ING-INF/03

Coordinatore: Prof. Felice Crupi

Firma Felice Crupi

Supervisore/Tutor: Ing. Floriano De Rango

Firma Floriano De Rango

Dottorando: Dott. Andrea Lupia

Firma Andrea Lupia

A Fabrizia, che mi ha sempre supportato durante questi 3 anni

Table of contents

List of figures

Chapter 1

Introduction

The security and the energy consumption are key aspects of distributed wireless ad-hoc networks. The major issues regarding security are due to the lack of an infrastructure and the use of the wireless medium for transmission and reception of data, therefore the communication could be easily disturbed or eavesdropped, or the nodes could be corrupted by malicious agents. The limited power source available to the nodes puts a severe constraint to the operations that a node can execute. This implies that any adopted measure to detect malicious behaviors must take into account the required energy, reducing it to the minimum, so the node lifetime will not be excessively affected.

In Chapter 2, an overview of various typologies of wireless ad-hoc network is given, analyzing what they have in common, and what distinguish them. The transmission technologies for wireless communication and their last enhancements are illustrated, analyzing the energy consumption they imply and the supported transmission data rates. The most known routing protocol for wireless ad-hoc networks published as RFC are described, eventually defining and categorizing the many threats to which they are subjected.

As countermeasure to these threats, the state of the art concerning the Intrusion Detection Systems on resource-constrained devices is depicted in Chapter 3. A common classification based on the techniques used by IDS to analyze the traffic is introduced, defining advantages and drawbacks of each technique. Main IDS architectures are hence described, defining the scenarios in which they perform better. Afterwards, many proposals in literature are illustrated, analyzing the most novel approaches for intrusion detection in wireless ad-hoc networks. Proposal are classified by the approach they use to detect intruders, with a special focus on trust-based detection, which will be exploited in the following of the thesis.

The analysis of energy consumption in MANETs using a TMS is shown in Chapter 4. Initially the trust model, the energy model and the attacker model are defined. They are

exploited to introduce a trust management procedure in a protocol that provides already encryption procedures, increasing the kind of attacks that the network can detect. Trust Management Scheme (TMS) provides protection against malicious nodes that fairly participate to the route establishment phase, dropping all or a part of the data packets received, in addition to the attacks already protected by encryption. Finally the performance evaluation is presented, by analyzing the results of the performed simulations, comparing the proposal with an encryption-based protocol.

In Chapter 5 a distributed time division-based monitoring strategy is proposed, to achieve the high security levels required while reducing the energy consumption. The proposal involves the concepts of trust and link duration, allowing the division of the monitoring task between trusted nodes, called companions. At the end, the analysis of simulation results is shown, allowing the evaluation of the proposal performance.

The proposal and analysis of an approach able to reduce the energy consumed during the monitoring operations is presented in Chapter 6. The main idea of the proposal concerns the reduction of the promiscuous mode usage, which is used for monitoring. The decision about performing monitoring activities is taken in a probabilistic way, depending on the trustworthiness of the chosen intermediate node. The approach is implemented in NS-3 simulator, therefore enabling the comparison with other known protocols. The results obtained through the simulations show an improvement in performance, especially in networks composed of just fair nodes.

Finally, the conclusions about the proposals are illustrated in Chapter 7, analyzing the obtained improvements and depicting the possible future works to further improve the described approaches.

Chapter 2

Security Threats in Wireless Ad-hoc Networks

Distributed wireless networks enable communication between hosts in various environments through the wireless medium. The ease of forming a network of this kind is achieved by not requiring an infrastructure, using multi-hop communication between nodes which are not in their respective range of communication. Networks like Mobile Ad-hoc Networks (MANETs) and Distributed Wireless Sensor Networks (WSNs) can be exploited when an infrastructure is too expensive or too difficult to set up, as in hostile environments or vehicular scenario. Some of the strong points of wireless ad-hoc networks are also their weaknesses. Some examples of threats are the following: communication through the wireless medium allows anyone in the range of communication to "overhear" exchanged messages between nodes; malicious agents could fake their identity or spread false information in the network; messages directed to a host could be misrouted or discarded. In this chapter an overview of main wireless ad-hoc networks, most common threats to them and some protection measures against them will be given.

2.1 Wireless Ad-hoc Networks

Wireless ad-hoc networks have many characteristics in common among them, but each type has some peculiarity that distinguish it from the others.

Fig. 2.1 An example of Mobile Ad-hoc Network scenario

2.1.1 Mobile Ad-hoc Networks

A MANET [1] is a network composed of nodes that can move freely in an area. These nodes can be in airplanes, ships, trucks, cars, even people or small devices. The network can be isolated or in communication with other networks through interfaces. Each node has one or more wireless antennas for transmission and reception of data, with antennas that can be directional or omnidirectional. Network topology changes over time, because it depends on nodes position, transmission/reception power of antennas and channel interferences. The main characteristics of MANETs are:

- dynamic network topology: nodes move arbitrarily, so network topology changes randomly and quickly, with multi-hop routes that need to be recreated when links between nodes break;
- reduced bit rate: mobile devices use low-consumption network interfaces because of their limited power (see below). New wireless standards (e.g. 802.11ac) have higher theoretical maximum, but wired connections achieve better results due to multiple accesses, interferences and signal attenuation;
- limited power: nodes in ad-hoc networks have no connection to the power line, because they are freely moving around an area. Therefore, any operation done by the node must take into account this constraint;
- low security of transmission medium: the access to a wireless network without security measures requires just to stay in transmission range of the communication, while wired networks need a physical connection to transmission medium. A node equipped with a wireless interface can eavesdrop exchanged data;

- high fault-tolerance: the distributed nature of MANETs allow to find routes that avoid faulty nodes without issues, because network protocols allow the dynamic discovery of new paths;
- network scalability: differently from wired networks, the amount of nodes participating in ad-hoc network can change often, so ad-hoc protocols quickly adapt when nodes join or leave the network.

2.1.2 Wireless Sensor Networks

WSNs [2] are composed of a large collection of sensor nodes organized as cooperative network. Each node is equipped with one or more sensors, memory storage, a battery and one or more processing units. Sensors are used to detect phenomena such as ambient light, heat, pressure, movement, noise. The absence of wires and infrastructures gives the opportunity of developing these networks in a simple way. Some of the applications of WSN technology concern smart grid, intelligent transportation systems and smart home. WSNs have many characteristics in common with MANETs, but they differ for some aspects:

- the amount of nodes composing a WSN are more than nodes of MANETs by some orders of magnitude;
- nodes density is higher;
- each node size is smaller, so nodes in WSNs have less energy, computational power and memory with respect to their counterpart in MANETs;
- the main purpose of sensor nodes is about the monitoring operations they perform, so a node and its sensor are strictly tied.

Data collected by sensor nodes is usually handled by each node taking part in multi-hop route followed by data, reducing the amount of traffic transmitted.

2.1.3 Vehicular Ad-hoc Networks

Vehicular Ad-hoc Networks (VANETs) [3] are a type of ad-hoc network that raised the interest of scientific community. The communication occurring between vehicles has some unique characteristics, such as the speed of the nodes, which can exceed 100 Km/h, so specific standard are needed to enable the communication in these networks. The most important field of application of these networks concern the safety while driving [4, 5], as

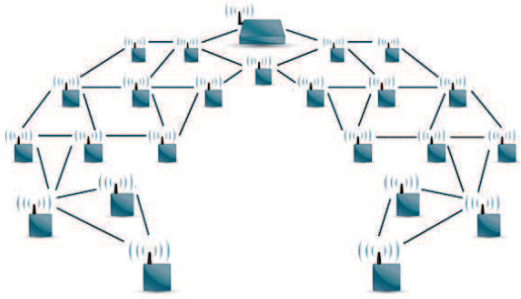


Fig. 2.2 An example of Wireless Sensor Network scenario

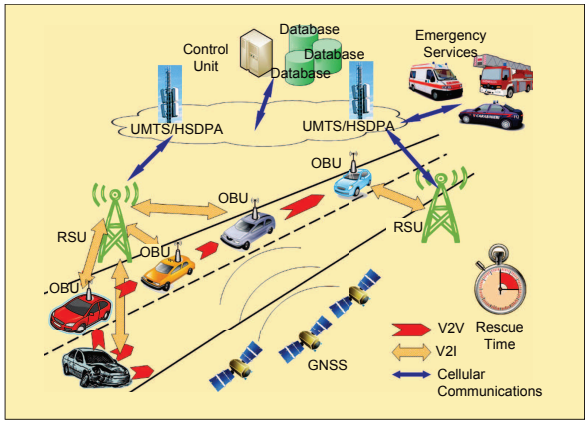


Fig. 2.3 A Vehicular Ad-hoc Network safety system

shown in Fig. 2.3. Their usefulness increases with the amount of vehicles provided with communication devices named On-Board Units (OBUs), so a high penetration rate of this technology is desirable to take advantage of their use. VANETs can be fully distributed, with car-to-car communication using OBUs, or an infrastructure can be provided along the roads, such as Road-Side Units (RSUs), which facilitate the communication among vehicles.

2.1.4 Flying Ad-hoc Networks

Concerning the Flying Ad-hoc Network (FANET) [6], it is a novel field that applies to Unmanned Aerial Vehicle (UAV), such as drones. Main characteristics of this kind of network are a higher mobility degree than nodes in MANET, because nodes fly in the sky. Therefore, high mobility brings to more frequent topology changes. Moreover, mobility is not influenced by the need of following roads. Distances between flying nodes are usually wider, but there are less obstacles interfering the communication. The communication between nodes in FANET is exploited also for coordination and collaboration between UAVs.

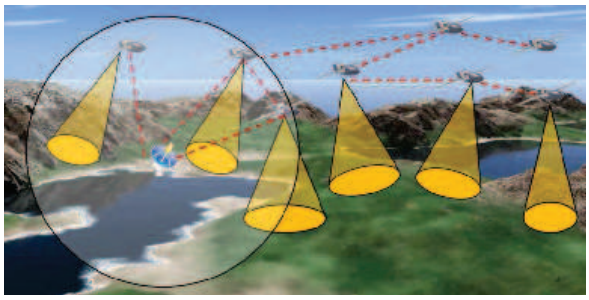


Fig. 2.4 An example of Flying Ad-hoc Network scenario

2.2 Transmission standards

Various technologies and standards provide wireless communication for many purposes. They differ for transmission range, energy consumption and maximum bit rate achievable, so they are used in different scenarios. In this section an overview of various wireless technologies is provided.

2.2.1 IEEE 802.11 standard

IEEE 802 is a family of standards for local and metropolitan networks. Protocols and services defined in these standards are about the two lower levels of Open System Interconnection (OSI) model: data link and physical layers. Data link layer is divided in two more sublayers, which are Logical Link Control (LLC) and Media Access Control (MAC). The working group 802.11 concerns Wireless LAN (WLAN) [7]. Various versions of 802.11 standard, developed and released since 1997, are briefly described as follows:

- 802.11 legacy: first version of the standard, released in 1997. The supported transmission bit rate is 1 or 2 Mbps, using 2.4 GHz band;
- 802.11a: released in 1999, this version uses 5 GHz band, allowing a maximum theoretical bit rate of 54 Mbps. The higher band with respect to the previous version makes the transmission more affected by obstacles;
- 802.11b: released in the same year of 'a' version, the band used by 'b' version of 802.11 is 2.4 GHz. It is compatible with first version of the standard with a higher bit rate (11 Mbps);
- 802.11g: using the same band of 'b' version and allowing a maximum bit rate of 54 Mbps, 802.11g was released in 2003. It can coexist with 'b' standard, with the

constraint that the maximum bit rate must be 11 Mbps is 'b' devices are connected to the network;

- 802.11n: become standard in 2009, it works using both 2.4 and 5 GHz bands. Its bandwidth is twice the bandwidth of previous versions (40 MHz vs 20 MHz), 'n' standard can reach a maximum bit rate of 600 Mbps exploiting Multiple-Input Multiple-Output (MIMO) technology;
- 802.11ac [8]: it is an amendment of previous version, released in 2013 and supporting wider channels (2 or 4 times wider than 40 Mbps) and Multi-user MIMO. It is the most recent standard adopted by commercial solutions as routers and laptops;
- 802.11ad [9]: promoted by Wireless Gigabit Alliance (WiGig) until 2013, it operates in the 2.4, 5 and 60 GHz bands, with maximum bit rate of 7 Gbps. At 60 GHz, signal cannot penetrate walls, but the propagation can occur by reflection;
- 802.11af [10]: this standard operates in TV white space spectrum, in VHF and UHF bands among 54 and 790 MHz. Using lower bands increases the possible range of transmission, suffering lower attenuation by obstacles;
- 802.11ah [11]: it uses bands under 1 GHz and takes advantage of lower energy consumption. Its field of application is the same of Bluetooth technology, providing low power consumption and wider range of transmission.

2.2.2 Bluetooth

Bluetooth technology [12] is used by mobile devices for short-range communication. Its first version was released by Ericsson in 1994, now its maintenance is managed by Bluetooth Special Interest Group (SIG). This protocol was designed to enable communication without using too much power. Based on transmission power, Bluetooth devices can be grouped in 4 classes, with higher number having less energy consumption:

- class 1, with maximum allowed power of 100 mW (20 dBm) and communication range of 100 meters;
- class 2, with maximum allowed power of 2.5 mW (4 dBm) and communication range of 10 meters;
- class 3, with maximum allowed power of 1 mW (0 dBm) and communication range of 1 meter;

- class 4, with maximum allowed power of 0.5 mW (-3 dBm) and communication range of 0.5 meters.

The frequencies in which Bluetooth operates are between 2400 and 2483.5 MHz, divided in 79 channels. Each channel is 1 MHz wide, with guard bands of 2 MHz and 3.5 MHz respectively at the bottom end and at the top. Network topology supported by Bluetooth is the scatternet, which can be composed of 2 or more piconets. A piconet is formed of maximum 8 devices (1 master and 7 slaves), with each device having a 3 bits address. Slave nodes in a scatternet participate to more piconets using time division multiplexing. The master node of a piconet can be slave in another.

With 4.0 version of Bluetooth, a low-energy version of the protocol was merged into the standard, with the name of Bluetooth Low Energy (BLE) [13]. Its improved consumption is due to lower latency and minimum time needed to send data, with halved peak current consumption (15 mA vs 30 mA) with respect to classic technology. Last released version of the standard is 4.2, with version 5 already announced.

2.2.3 ZigBee

A low-power protocol for communication is ZigBee [14], based on IEEE 802.15.4 specifications for physical and MAC layers. It was mainly designed for sensors and control devices, which require short-range low-rate wireless data transfer. It operates in Industrial, Scientific and Medical (ISM) radio bands, supporting bit rates from 20 to 250 kbps. ZigBee provides facilities for establishing secure communications through the use of symmetric 128-bit cryptographic keys.

2.2.4 LTE and 5G

Also known as 4G, Long-Term Evolution (LTE) is a cellular network standard for high-speed wireless communication [15]. It uses the bands of 800 MHz, 900 MHz, 1800 MHz and 2600 MHz in Europe, while in the United States the bands it uses are 700 MHz and 1700 MHz. The maximum theoretical transmission bit rates achievable by LTE is 326.4 Mbps in download and 86.4 Mbps in upload, a high improvement with respect to the last version of the previous technology HSPA (42 and 11 Mbps respectively). The width of the exploited channel is variable (from 1.25 to 20 MHz), providing a better scalability than previous generation standard.

The 5th generation mobile networks, known as 5G, are proposed as next telecommunications standard [16]. For these networks, requirements to satisfy are defined, as data rates of 100

Fig. 2.5 5G standard logo

Mbps in metropolitan areas, management of more than 100000 simultaneous connections for massive WSNs, improved coverage, enhancement of signaling efficiency and more. The expected roll out date of this new standard is the year 2020.

2.3 Routing protocols

Ad-hoc networks are characterized from high nodes mobility. A routing protocol has to keep track of the topology changes. Nodes need to collaborate through exchanging information, therefore a protocol supports all the operations needed to manage the data they send to each other. During route discovery phase, this route should be computed avoiding loops. Depending on the need of saving bandwidth and other network resources, or having always fresh information about routes toward other nodes, a routing protocol can have a different approach to spread information. A commonly used classification for ad-hoc routing protocols [17] consists in 3 categories:

- proactive;
- on-demand (or reactive);
- hybrid (both proactive and on-demand operations).

Proactive protocols keep updated information for every route by exchanging data even when the nodes do not need a path for a destination. Routing information is stored in tables by each node, which reacts to topology changes by spreading the updates and refreshing the information in the routing table. The disadvantage of this approach concerns the constant use of part of the bandwidth to exchange overhead data.

A more dynamic approach is used in on-demand protocols. Nodes start a route discovery process only when they need a path toward a destination. The discovery ends when

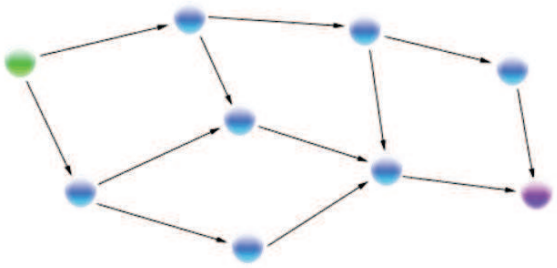


Fig. 2.6 RREQ propagation in AODV protocol

the destination node (or a node that knows a route to reach it) reply to a request. This approach allows the saving of bandwidth and energy, at the cost of higher delays for the first transmission.

2.3.1 Ad-hoc On-demand Distance Vector protocol

Ad-hoc On-demand Distance Vector (AODV) protocol [18] belongs to reactive protocols, allowing the nodes in the network to quickly establish the routes toward new destinations, without maintaining unused paths. The main feature of this protocol is the sequence numbers. Each node keeps a number that is used to know if a route is more recent than another, avoiding the creation of loops while discovering a new route.

A node that wants to send data toward a new destination, for which it does not already have a valid route, starts a new route discovery process. It starts by broadcasting a Route Request (RREQ) packet, containing the information about the originator node (i.e. the node which needs the route) and the destination node. This packet is retransmitted in broadcast by the nodes which receive it until it reaches a node that knows a route to a destination, or the destination itself. The retransmission of the same RREQ packet is avoided by including a RREQ ID in it. The protocol exploits the expanding ring search for trying to avoid flooding the whole network when the destination node is close to the originator node. The Time to Live (TTL) starts from a low value, increasing each time there is no reply and the originator node sends again the request.

When a node is the destination of a request, or it knows the route to the destination, it generates and sends a Route Reply (RREP) packet. It is sent in unicast toward the originator, following the reverse path created while broadcasting the request.

When a link break occurs, the node using that link in a route transmits the information about the link toward the interested nodes through the generation of a Route Error (RERR) packet. It contains the unreachable destination, so the nodes receiving the packet can update their routing table by marking the route as invalid, and starting a new route discovery process if they still need to communicate with that destination.

Nodes can maintain the connection with their neighbors by optionally using Hello messages. They consist in messages periodically broadcast with TTL equals to 1, so a node can notice which nodes are still in its transmission range.

A new version of this protocol [19], previously known as Dynamic MANET On-demand (DYMO) and now as AODVv2, was in draft version until the end of November 2016, without becoming standard for Internet Engineering Task Force (IETF). With respect to AODV first version, it does not provide support for Hello messages and local repair (a technique to locally find an alternative to a route with a broken link). A new feature of the protocol concerns a mechanism for using multiple metric types.

2.3.2 Optimized Link State Routing protocol

Optimized Link State Routing (OLSR) protocol [20] is a proactive protocol. It is an adjustment of the link state algorithm for ad-hoc networks. The key concept of this protocol is is the Multi Point Relay (MPR). They are nodes that are chosen to forward broadcast messages during flooding process. This technique allows to reduce the overhead with respect to the classic flooding mechanism, where each node forwards the first received copy of the message. In OLSR protocol, information about links state are generated and transmitted just by MPRs. Moreover, this protocol allows sharing partial information about links state.

An updated version of this protocol, OLSRv2 [21], was published as standard in 2014. It keeps the same base mechanisms of the previous version. Main improvements concern the link metric, with the support extended to other metrics than the hop count, and a more flexible and efficient signaling framework. Moreover, protocol messages are simplified.

2.3.3 Destination-Sequenced Distance Vector protocol

Another proactive protocol is Destination-Sequenced Distance Vector (DSDV) [22]. It is based on Bellman-Ford algorithm. Each node keeps a routing table where all destinations in the network are registered. The amount of nodes needed to reach each destination is saved in the table. Similarly to AODV protocol, the freshness of a route can be defined through the sequence number of the destination, avoiding the creation of loops too. Updates of routing

table are transmitted at a fixed time interval, maintaining its consistency. Overhead can be reduced transmitting just incremental information of the routing table when possible.

2.3.4 Dynamic Source Routing protocol

Dynamic Source Routing (DSR) [23] is a reactive protocol, based on source routing. The route toward a destination is discovered by broadcasting a route request. Each node that receives it for the first time checks if this route is in its route cache. If it does not contain the requested route, the node will add its address in the route record of the packet and broadcasts it again. The discovery ends when the request reaches the destination or a node with a valid route to it. The reply message contains the route record of the corresponding request, with the route cache concerning the destination in addition if the reply is generated by an intermediate node. The reply follows the route record in reverse order.

Routes are maintained by using route error packets, generated when layer 2 detects a transmission problem when communicating with a neighbor node. Route error packet triggers the deletion of the unreachable node and all the routes containing it from the route cache of the node that receives it.

2.4 Security attacks

Ad-hoc networks are subject to different kinds of threats, many more than their wired counterpart. Mobility and lack of infrastructure raise many issues under the security perspective.

2.4.1 Security goals

One of the approaches in securing ad-hoc networks [24] defines the attributes to satisfy to address the network security:

- availability;
- confidentiality;
- integrity;
- authentication;
- non-repudiation.

Achieving these goals should lead to a secure network, strong against malicious behaviors.

Availability

The availability is intended as the opportunity of using a service offered by a node when needed. In ad-hoc networks, a node could interact with another one exploiting allowed operations but with malicious intents. Actions against the availability are known as Denial of Service (DoS) attacks. In networks composed of nodes with limited power sources, the main aim of a DoS attack could be the depletion of nodes energy, having worse consequences than in other contexts.

Confidentiality

The impossibility of accessing to some information by unauthorized entities is known as confidentiality. The principal method used to achieve this goal is the encryption of confidential information. This property is fundamental in some scenarios to avoid the eavesdropping of sensitive information. Prerequisite of this property is the authentication, because securing information is useless if the identity of the sender or the receiver cannot be ensured.

Integrity

Integrity as requisite consists in preventing or at least detecting the unauthorized modification of a message. It could depend of the behavior of a malicious node, which wants to modify the content of a message for its purposed, or to the interferences in the wireless medium.

Authentication

The authentication allows to acknowledge the identity of an agent. Without this property, a node cannot be sure about the node with which it exchanges information, so a malicious agent can access data intended for other nodes and interfere with operations on behalf of another entity. Authentication could be managed in many ways, e.g. through key exchange mechanisms or Public Key Infrastructures (PKIs).

Non-repudiation

Non-repudiation attribute states that a node cannot deny the authorship of a message. With this property, the author of an erroneous message can be accused without doubts, and the

information can be spread to other nodes. A common method to achieve non-repudiation is the mechanism of digital signatures.

2.4.2 Internal attacks

Nodes composing an ad-hoc network could be placed in a hostile environment, with weak physical protections. These scenarios put nodes in danger, because they could be compromised, consequently acting in malicious way. Requiring a centralized entity, as in WSNs, is source of vulnerabilities, because compromising just this entity could lead to the entire network disruption.

2.4.3 External attacks

Using wireless connection between nodes could expose the network to active and passive attacks. Active attacks can be the modification or discarding of messages circulating in the network, forging of fake messages, impersonation of another nodes with the purpose of disrupting the connection between nodes. As passive attacks, the eavesdropping could allow the access to reserved information by malicious agents.

2.4.4 Attacks categorization

In literature many ways to categorize attacks were proposed [25]. One possible categorization divides the attacks in two typologies: route-disruption and resource-consumption. Attacks attempting to manipulate route messages with the aim of disrupting routes between nodes are part of the first typology, while resource-consumption attacks try to consume energy, bandwidth or storage memory of the nodes in the network by transmitting fake or wrong packets.

Another categorization divides attacks in three types: modification, impersonation and fabrication. Modification consists in altering routing packets content, modifying information contained in its fields. A node can impersonate one or many nodes, redirecting the traffic directed to it or generating loops in routes. Fabrication attacks are made by generating fake packets, containing wrong information to break links in the network, consuming the nodes energy and the network bandwidth. Malicious nodes can also take advantage of trust-based Intrusion Detection System (IDS), spreading false information about other nodes. They can use high trust values to recommend other malicious nodes, or low trust values for nodes behaving correctly.

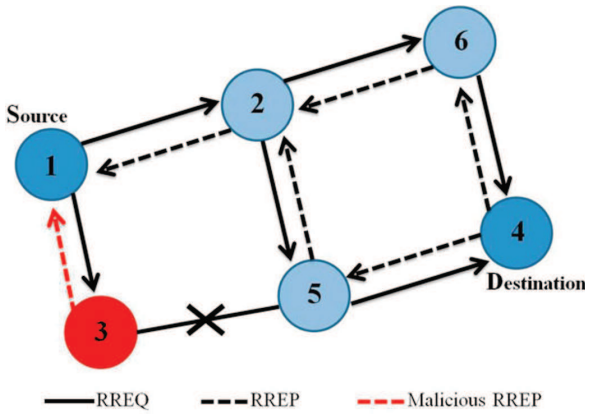


Fig. 2.7 Malicious node (in red) performing black hole attack

A survey done on MANET intrusion detection and prevention approaches [26] categorizes various attacks in two categories: active and passive attacks. The first category refers to attacks actively performed by malicious agents, as sleep deprivation, black hole, sybil attack and more. Passive attacks category includes eavesdropping, traffic analysis and location disclosure, which are attacks that do not directly affect the functionalities of the network, but they can be dangerous in some scenarios.

Sleep deprivation

The sleep deprivation attack can consist in sending route requests for a node unavailable in the network or sending many requests without waiting any time between them. It is a distributed DoS attack, where a node interacts with one or more nodes in a way that seems legitimate, but with the aim of depleting the energy of the attack targets. As the energy is a strict constraint in ad-hoc networks, this attack is very effective in this scenario.

Black hole attack

A malicious node performing a black hole attack [27, 28] tries to redirect the most of the traffic in the network through itself, then it drops all the data packets that it receives, as shown in Fig. 2.7. The attraction is performed by exploiting protocol packets and the information about the metric. The malicious agent advertises the path with the least cost, so nodes will choose the route including it. Higher is the amount of routes in which the malicious agent is included, higher is the effect of the black hole attack against the network.

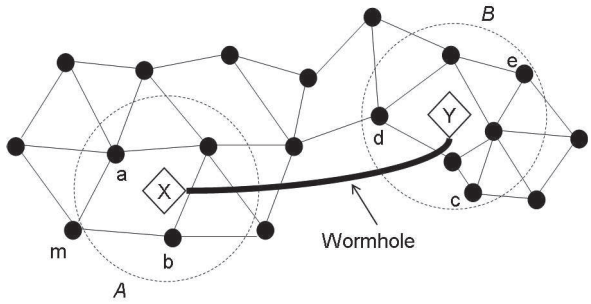


Fig. 2.8 Wormhole attack

Gray hole

The gray hole attack [28–30] is very similar to the black hole attack, but it does not drop all the packets. Based on the data contained in the packet or in a probabilistic way, just a part of the packets is forwarded. Usually this attack is harder to detect, because in some circumstances fair nodes cannot forward packets (e.g. when a link breaks due to node mobility), so nodes can imagine that this behavior is not malicious.

Packet dropping

A variation of black or gray hole attacks is the simple dropping of the packets. This threat is performed without trying to attract the routes by malicious agent, but just dropping all or a part of the packets to forward. This behavior can be also not driven by malicious intents, because in ad-hoc networks there are situations that makes impossible forwarding a packet (e.g. the unavailability of the next hop node because it moved from its previous position).

Wormhole

The wormhole attack [31] is an active attack, consisting in tunneling the packets received from a point of the network to another. For tunneled distances longer than the wireless range, the packet sent by the malicious node arrives before the packet that follows the multi-hop route, so the malicious node gains a powerful position in the network. This attack can be performed by using a wired connection or directive antennas. Malicious agent can forward data intended for it, or all the packets eavesdropped in its transmission range. Data can be tunneled bitwise, reducing even more the time needed for the transmission. In this way, a malicious node can be part of many routes. The malicious behavior in this attack is intended as the lack of retransmission of some packets after the malicious agent obtained a privileged status in the network, being part of most of the routes.

Rushing attack

An effective active attack against reactive routing protocols for ad-hoc networks is the rushing attack [32], which consists in spreading the route requests quickly or advising them as the latest ones. Therefore, the malicious node takes part of the routes, because to control the packet overhead, only the first request is taken into account by other nodes.

Sybil attack

The sybil attack [33] consists in exploiting the lack of central authorities for identity verification, typical in ad-hoc network, to fake other identities and send control packets on behalf of other unaware nodes. Therefore, all the packets directed to a node will be sent to the malicious node.

Byzantine attack

The byzantine attack [34] requires the cooperation between malicious nodes. They can jointly perform one of other described attacks, or they can create loops between them. Therefore, they seem to have a fair behavior from other nodes perspective, and their detection is very hard.

Eavesdropping

Using the wireless medium to communicate can be exploited by malicious agents to hear data transmitted. A node undergoing this attack is not aware of what is occurring. In some scenarios communication is private, so countermeasures as the encryption need to be adopted.

Traffic analysis and location disclosure

The analysis of the traffic exchanged by nodes in a wireless ad-hoc network can be exploited to discover their location. This attack can also work without taking into account the content of messages, but just analyzing the communication pattern, the amount of data transmitted and the transmission characteristics.

Chapter 3

Intrusion Detection Systems on Resource-Constrained Devices

Intrusion detection can be defined as the automatic detection and the subsequent alarm generation of an intrusion that is being performed (or it was already made). An IDS is a protection system capable to detect hostile and malicious activities that put the network in danger. Intrusions are detected by monitoring the traffic on the network, looking for suspect activities. After the detection, it should take the adequate countermeasure to protect the network. The main purpose of an IDS is to offer a second line of protection, because it intervenes when one or more agents are already compromised.

3.1 IDS classification

An IDS can work by analyzing sent and received traffic of the network interface or analyzing log files stored on the device. The main techniques used by IDSs [35, 36] for detection can be divided in three categories:

- anomaly detection systems;
- misuse detection systems;
- specification-based detection systems;
- hybrid detection systems;

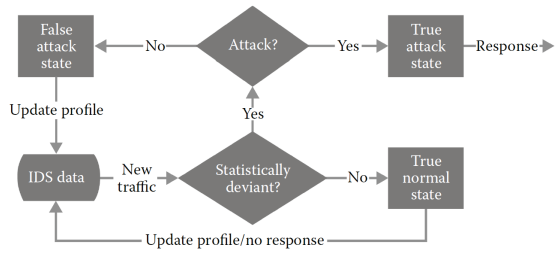


Fig. 3.1 Anomaly detection process flow

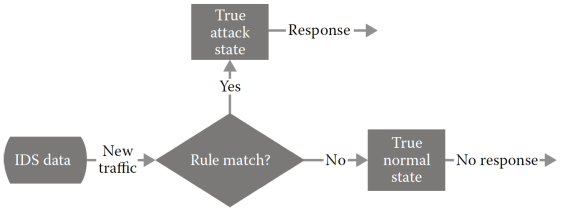


Fig. 3.2 Misuse detection process flow

3.1.1 Anomaly detection systems

The anomaly detection is performed by comparing normal behaviors with collected data. The activities are considered malicious if they deviates enough from the normality. Abnormal traffic is different than normal traffic, moreover it is less common. With these premises, this type of detection should be capable of identifying new attacks, because unknown attacks deviate from normality too. Variables taken into account by the IDS are a finite number, so it could lead to false positives , that is fair traffic detected as malicious, and false negatives, meaning that malicious traffic is not detected.

3.1.2 Misuse detection systems

As misuse detection, the analysis of the patterns and signatures of known attacks is used for comparison with collected data. If there is a match, it is treated as an intrusion. The main difference with respect to the anomaly detection concerns the patterns used for detections, because they are defined off-line by an expert. Based on this observation, the detection following this concept works very well when the attack is already known. In opposition to this, novel attacks cannot be detected at all. Another drawback of this approach is given by the definition of the patterns from the historical data, therefore known attacks tend to be less used and the rules become outdated.

3.1.3 Specification-based detection systems

The detection based on specifications allows to detect as malicious a monitored behavior that differs from a set of defined constraints. The main assumption concerns fair nodes, which will behave within the defined bounds. Activities are not marked as common and uncommon, rather they are identified as what a system may and may not do. As in misuse detection systems, the correctness and completeness of the specifications depend on the expert knowledge, mostly the second one, because generating all the specifications for the amount of programs used today is a very hard task.

3.1.4 Hybrid detection systems

Combining previous described systems could improve the intrusion detection, using each approach for its advantages. Obviously, the way in which two or more systems are joint together will determine the performance of the IDS.

3.2 IDS architectures

The optimal architecture for an IDS applied to wireless ad-hoc networks depends on the role of the nodes for routing reasons. If all the nodes have the same tasks, then they could make the same operations for monitoring and detection of malicious agents. If the nodes participating in the network are divided following some criteria (e.g. forming clusters), then nodes with a broader view of the network should have a different role in intrusion detection. The main architectures for intrusion detection [37] are:

- stand-alone IDS;
- distributed/cooperative IDS;
- hierarchical IDS.

Each architecture has its advantages and drawbacks. In stand-alone IDS, each node has the task of monitoring and detecting threats for itself. All the information is collected locally, all the decisions are taken by the node independently from the others. The node does not know if other nodes have taken any decision about an agent, because the nodes do not exchange alerts or recommendations. If all the nodes in the network are capable of running an IDS, this architecture can be suitable for wireless ad-hoc networks.

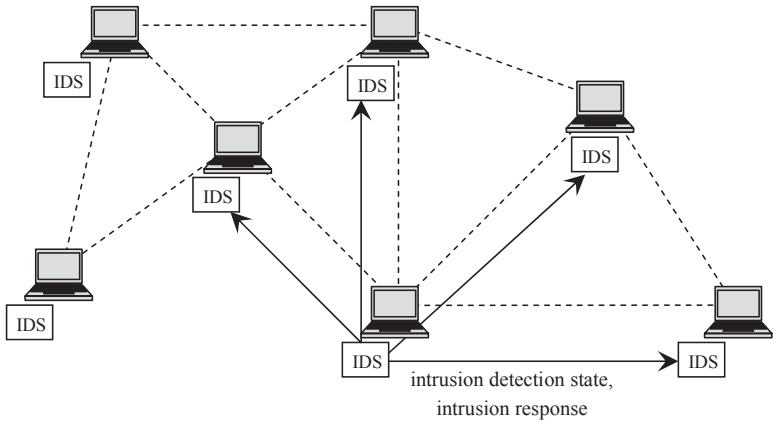


Fig. 3.3 Example of distributed and cooperative IDS

As the nodes cooperate for achieving multi-hop transmission of data through the network, they can also collaborate for intrusion detection. Every node participates in this system by collecting data about possible intrusions and sharing it with the rest of the network, as shown in Fig. 3.3. The decision about the maliciousness of a node is taken in a distributed way, based on the information gathered by all nodes involved in this process.

For hierarchically organized networks, an IDS that takes into account this characteristic is suitable. Nodes that perform more actions (e.g. cluster-heads) can act as control points. Each node can take decisions locally, while nodes with more assigned tasks can start global actions against an intruder when it is detected.

3.3 Detection approaches

The way in which the behavior of the nodes is evaluated characterizes IDSs. The analysis of many recent works led to the following approaches:

- trust-based;
- game theory-based;
- machine learning-based.

Despite the encryption is not a proper method of intrusion detection, because it is more a method to prevent the intrusions, a brief analysis of various encryption-based approaches is also presented in this chapter.

3.3.1 Trust-based detection

Many approaches in intrusion detections are trust-based [38, 39]. There are many ways to assign a trust value to each node. Trust values can be exploited as metric, or just to exclude malicious nodes from the routes. Nodes can evaluate the trust of an agent by the data gathered during the interactions with it, in a direct way, or collaborating with other nodes by asking and/or receiving information about an unknown agent.

The authors in [40] propose an approach to improve detection accuracy through Collaborative Intrusion Detection Networks (CIDNs). They introduce the concept of intrusion sensitivity to describe the accuracy of a certain IDS in detecting a specific type of attack. The key components of the CIDN framework used are:

- IDS nodes;
- trust management component;
- query component;
- collaboration component;
- communication component.

Each node running the IDS is referred as IDS node. It can collaborate with others, keeping a *partner list* that contains them. Joining the collaborative network requires getting a proof of identify from a trusted Certification Authority (CA). The trustworthiness of the nodes is evaluated through the trust management component. The authors take into account two different types of trust:

- feedback-based trust (also known as indirect trust), which is established through the collaboration with the *partner* nodes using the collaboration component;
- packet-based trust (or direct trust), computed by evaluating the validity of received packets from the target node.

The query component enables a node to send a set of queries to a target node containing a set of alarms, receiving answers that depend on the configuration and the settings of the target node IDS. Computing feedback-based trust requires the use of the collaboration component to exchange requests and challenges between nodes, in order to receive the corresponding feedback. The requests enable the consultation of the alerts, challenges are used to evaluate the trustworthiness of *partner* nodes. Replying to these operations requires the sending of a

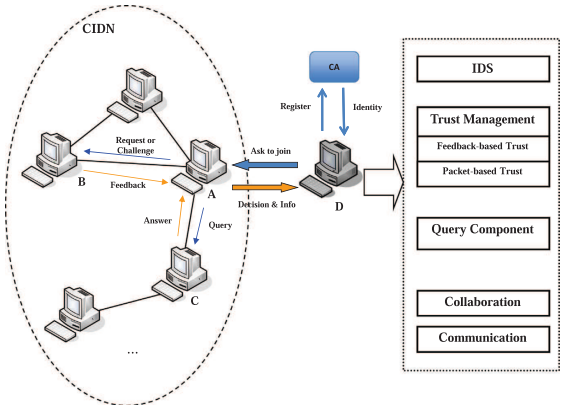


Fig. 3.4 CIDN framework

feedback. The connection between the various IDS equipped by nodes is maintained through the communication component, which can also help a node in computing the packet-based trust. Fig. 3.4 illustrates how CIDN works.

A trust-based mechanism to secure OLSR protocol is proposed in [41]. Nodes trustworthiness is evaluated through a fuzzy Petri net. It is a combination of classical Petri net with fuzzy logic. The trust evaluation is done by applying fuzzy rules, in the form of "IF x IS $property_x$, THEN y IS $property_y$, which are two fuzzy propositions. Each proposition has a credibility called truth degree. A Petri net is composed of transitions, places, tokens and arcs. Places and transitions are connected by arcs, while tokens can be contained only in places. The reasoning about trustworthiness is made by mapping entities of Petri net with fuzzy logic. Places are propositions, transitions are mapped as casual relationships of propositions and tokens represent the trust degree of a proposition. Tokens can take values in the continuous interval from 0 to 1. The OLSR protocol is enhanced using a trust based routing algorithm that selects the path with maximum trust value among all the possible paths. The trustworthiness of a path is evaluated as the minimum trust value of the nodes included in it. This choice is made possible by the type of protocol, OLSR, which is a proactive protocol, therefore nodes have information about the whole network topology.

The work in [42] concerns the development and the analysis of a trust management protocol for MANETs using hierarchical modeling techniques based on Stochastic Petri Nets (SPNs). The concept of "web of trust" is used in order to extend the trust over the space based on a weighted transitivity of trust. The obtained degree of trust is based on the length of the trust chain. Longer trust chains mean a higher decay of the trust degree. A node can recommend only agents with which it had previous interactions. The trust metric exploited

by authors in this proposal takes into account quality of service and social trust aspects. The nodes behavior is described by a hierarchical SPN model.

An overview of TMS applied to MANETs is offered in [43]. In this survey, authors emphasize the important phases composing the trust management, such as trust establishment, trust update and trust revocation, which can severely affect the network performance if they are not harmonized in the specific routing protocols. On the basis of the routing scheme applied, these TMSs need to be adapted to perform well without degrading the overall network performance. Authors emphasize, as for future directions, TMS with IDS should be able to trade-off among more metrics and resources such as time or energy, especially in MANETs.

The authors in [44] addressed the performance issue of trust management in MANETs for trust bias minimization and application performance maximization. The trust management protocol developed by authors adopts a combined metric, which uses social and Quality of Service (QoS) trust. Social trust is evaluated through social ties and honesty, measured respectively by intimacy and healthiness. The evaluation of QoS trust is done taking into account the capability of a node to complete a mission assigned. The chosen metrics are the energy and the cooperativeness in protocol execution, which represent the competence and the protocol compliance. The trust value of a node is represented by a real number between 0 and 1, with 0 meaning complete distrust, 1 for complete trust, and the 0.5 value representing a state of ignorance. The chosen metrics are explained in the following:

- intimacy measures the interactions between nodes, concerning packet routing and forwarding;
- healthiness represents the belief about the node fairness, therefore it is related to the probability that a node is compromised;
- energy is referred to the residual energy of a node, because in MANETs the energy is limited by the battery capacity, so the capability of completing a task depends on it;
- cooperativeness is intended as the participation of a node in routing operations, such as routing and packet forwarding.

In addition to the healthiness, monitoring is exploited in order to evaluate the trustworthiness of a node. The behavior of fair, malicious and selfish nodes is modeled using SPN techniques. The adopted model is shown in Fig. 3.5 The work introduced the concept of objective trust evaluation, using knowledge concerning the environment conditions.

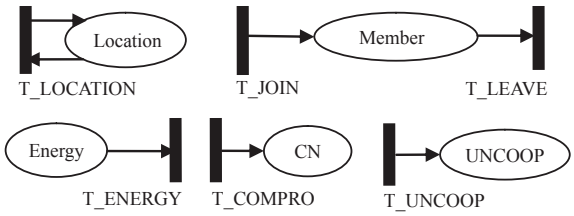


Fig. 3.5 Node SPN model

A proposal of a probabilistic detection scheme for Delay Tolerant Networks (DTNs) is presented in [45] with the name of iTrust. Some particular wireless ad-hoc networks are included in this type of networks (e.g. WSNs with scheduled intermittent connectivity, sparse MANETs). The iTrust scheme is inspired from the inspection game, a game theory model in which the adherence to certain legal rules of an entity, called inspectee, is verified by another entity, known as inspector. The scheme provides a Trust Authority (TA) that is periodically available. Its behavior follows a probability, based on which it could launch the probabilistic detection by collecting history evidence of a target node to use the information when it will be judged. The reputation system introduced by the proposal provides lower probability of controlling a node with a good reputation. The iTrust scheme analysis is done using the game theory, in order to demonstrate that TA could provide security for DTN routing. The proposal is evaluated against selfish and malicious nodes performing black hole and gray hole attacks. Evidences evaluated for judgment are the following:

- delegation task evidence: used to record the number of tasks about routing assigned to a certain target node;
- forwarding history evidence: it is verified with a signature provided by the intermediate node to which the packet was forwarded. The target node submits its history to TA, which evaluates it taking into account the tasks delegated;
- contact history evidence: each time two nodes have a contact, a new evidence is generated and store in both nodes.

The probabilistic approach is introduced to reduce the cost of evidence evaluation in an advanced version of iTrust. The Nash equilibrium of the inspection game is given by a mixed strategy, with positive probabilities of inspection and non-compliance. The analysis of the results of various experiment performed showed a reduction of the overhead generated by the IDS.

The work in [46] proposed a trust prediction model in which each node computes historical trust of its neighbors using the packet correct forwarding ratio. The proposed model classifies the trust in three different types:

- node historical trust;
- node current trust;
- route trust.

Node historical trust is computed using the information about the past direct interactions with the node. It is composed of two factors: control packet forwarding ratio and data packet forwarding ratio. Each one has a different weight to determine the overall historical trust. The adopted mechanism to evaluate interactions is the monitoring of transmitted packets through the promiscuous mode of the wireless interface. As node current trust, the application of the fuzzy logic rules to the historical trust is intended. Current trust predicts the future behavior of the subject node. The route trust is exploited for evaluating the quality of a chosen route. The trust value of a route depends on the trust values of the intermediate nodes along it. The proposed protocol, named Trust-based Source Routing (TSR), extends the source routing mechanism with the trust model just described. The route is chosen taking into account the minimum hop count among the routes that satisfy a defined trust requirement.

The authors in [47] proposed a trust model which evaluates neighbors direct trust using these attributes: time of encounter, mobility and successful cooperation frequency. The recommended trust value is determined using the revised D-S evidence theory. This theory is based on the identification frame Ω set, containing basic propositions which are both exclusive and exhaustive. Then Ω is defined as $T, -T$, with T and $-T$ representing respectively the credible and the incredible states. From the set 2^{Ω}, which represents the set of all the possible propositions based on Ω, two concepts are defined: belief and plausibility. The different between belief and plausibility is defined as belief interval, which is the range of maximum uncertainty. The trusted routing protocol TDS-AODV is based on the proposed novel trust mechanism. It is an extension of the AODV protocol, with each node making a routing decision according to the trust values of its neighbor nodes. The protocol provides the possibility of building 2 different routes: the main route with highest trust value among all the candidate routes, and a backup route.

A proposal of trust-based Secure AODV (SAODV) protocol with intrusion detection and incentive cooperation was discussed and analyzed in [48–50]. In this work, the protocol exploited an Intrusion Detection Mechanism (IDM) and a Trust-Based Mechanism (TBM)

to promote collaboration among nodes, penalizing selfish and malicious nodes. TBM stores useful information in a supplementary table, as follows:

- $trust\_id$: the address of neighbor node;
- N_{RREQ}: the number of RREQ correctly received;
- N_{RREP}: the number of RREP correctly received;
- T: the trust level, with 1 as initial value;
- N_f: the number of consecutive messages whose verification is failed. This value is exploited to reduce the trust level of an agent;
- th_{N_f}: maximum number of signature failures before putting the trust level to 0, temporally excluding the node from the communication.

Selfish nodes are detected through TBM and a credit management to promote cooperation, which takes into account the generation and the forwarding of control packets to increase and decrease the amount of credits.

In the work presented in [51], the authors proposed a trust based protocol for energy-efficient routing in MANETs. The main concept introduced by this proposal is the Energy-Factor (EF) computed through an energy consumption model. It is the ratio of the residual energy to the initial energy of a node, adopted by the proposed Protocol for Energy-Efficient Routing (PEER) as routing metric. The trust evaluation process is performed in three phases, picked from the five phases proposed in [52]:

- initial phase: the trust module initializes the EF of each new node joining the network;
- update phase: in this phase the trust values of the nodes that leave and re-enter in a node transmission range are tracked;
- re-establish phase: the possibility of misbehaviors due to mobility and low energy available are taken into account, providing a redemption mechanism that re-establish the trust value of a selfish node in this phase.

The proposed protocol sets up the routes using the values provided by the trust module, introducing some modifications in standard RREQ and RREP packets used by AODV protocol.

3.3.2 Game theory-based detection

Many proposals use game theory models to secure ad-hoc networks. A survey on latest trend is presented in [53]. Approaches can be categorized in non-cooperative games and cooperation enforcement games.

Non-cooperative games

Authors in [54] proposed a proactive defense scheme using an evolutionary game theory model. Each node tries to find the best strategy to balance its own rewards in terms of forwarding of data packets and energy consumption. The defense strategy of the nodes is dynamically adapted to attackers strategies.

Another approach is defined through an energy aware Trust Derivation Dilemma Game (TDDG) [55]. The work presented a risk strategy model to promote nodes cooperation, then the TDDG is introduced. Authors discussed the optimal ratio between the gain in terms of security, the cost in terms of energy consumption and the probability of the selected strategy.

Authors in [56] proposed an adaptive coordinator selection algorithm in order to secure the network against attacks and reduce the transmission delay. The game model on which the algorithm is based consists in a stochastic game for dynamic defense and an evolutionary game for coordination selection. The maximum payoff for players is obtained combining the Nash equilibrium strategies of both evolutionary and stochastic games.

A method to study an optimal monitor placement for IDS is given in [57]. The problem is modeled as a two-player zero-sum finite stochastic game between the attacker and the defender, which is the IDS. A target node can be in one of these two states: healthy or compromised. A node in compromised state means that it is controlled by an attacked, then it can inject malicious packets in the network with the main of attacking healthy nodes. The equilibrium of the game is characterized by analyzing attacking and defending strategies. The best results are obtained when both attacker and defender know the state of the network.

Cooperation enforcement games

An approach for a fair energy consumption distribution among nodes in a hierarchical-cluster network is proposed through the Trustworthy Energy Efficient Routing (TEER) algorithm [58]. The game theory is applied in cluster-head election. The cluster-head with higher energy and trust level corresponds to the Nash equilibrium of the game.

The authors in [59] analyzed the impact of the game theory on many network attributes, such as throughput, battery consumption and detection accuracy. Each node decides to

forward packets by defining both a cost and a profit for routing and forwarding packets, and keeping a history of interactions with non-cooperating nodes to exclude selfish node from the network. The incentive for nodes is keeping a good reputation, so they need to find a trade-off among maintaining their reputation and saving energy.

The application of game theory and fuzzy Q-learning to detect Distributed DoS (DDoS) attacks in WSNs is proposed in [60]. The Game-Fuzzy Q-Learning (G-FQL) is defined as a game with three players: a cluster-head, the sink and an attacker. The game is composed of two phases: in the first phase, the fuzzy Q-learning algorithm is used to check if the attack level of the presumed attacker is above a threshold. In this case, an alarm is transmitted to the sink, which prepares a countermeasure strategy in the second phase.

3.3.3 Machine learning-based detection

Machine learning can be exploited to detect malicious behaviors, mainly through the application of classification algorithms.

A proposal for intelligent intrusion detection in WSNs is given in [61]. The detection of anomalies is done through an IDM based on Random Neural Networks (RNNs). The solution was implemented on Arduino boards and compared with a encryption-based system, against the attack of a malicious node that transmits invalid data to degrade performance of the network. The evaluation of the proposal showed an energy saving of about 10% higher with respect to the energy consumption of the encryption-based IDS.

The Hybrid IDS (HIDS) is proposed in [62]. It is an anomaly based detection system based on Support Vector Machine (SVM) technique. It is a class of machine learning algorithm used for the classification of small sample data. In the off-line training phase, the system gathers and processes data from the physical, MAC and network layers. Then a mapping procedure is performed, classifying the training data through a division of the classification hyperplane by a linear classification plane. The discovery of malicious nodes is done through a signature based model, using a set of rules to classify the behavior of an agent.

3.3.4 Encryption-based prevention

The encryption-based approach is not a proper IDS, but it prevents the intrusions. Therefore, proposals using this approach can be integrated with IDSs, offering a more robust security.

SAODV [24, 63] is a routing protocol for MANETs based on AODV. Its principal task is securing the route discovery process. The major vulnerabilities of AODV that SAODV solves [64] are the following:

- impersonation made by a node that generates requests in name of another node;
- decreasing the hop count or increasing the sequence number of the destination by an intermediate node, to be part of the path connecting two nodes. This could happen if the node wants to analyze the packets exchanged by the nodes, or to break the link by dropping packets;
- impersonation of a node, generating a RREP with its address as destination address;
- generation and transmission of a RERR packet, impersonating another node. Using a high sequence number, next route discoveries regarding that node will fail;
- impersonation of a node generating and transmitting a RREP packet, declaring that the node is the destination and it is the leader of a subnet. Doing so, the node could drop all the packet of the subnet;
- sending of route request packet using the maximum sequence number possible for the destination. When it happens, the sequence number will start again from zero, so it invalidates all previous routes discovered.

The primary security requirement that SAODV satisfies is the import authorization, which is the authorization to update routing information only when the information is received by the destination itself. It needs other security services, such as integrity and source authentication. Integrity ensures that the message information was not modified by intermediate nodes, while source authentication is needed to verify that the node is who claims to be. These properties combined define the data authentication. They are achieved with digital signatures and message authentication techniques. The field integrity protection is obtained by using digital signatures to secure the packet. In this way, the fields cannot be modified by any node except the one that generates the packet. The only field not involved in this process is the hop count field, because each node that forwards the packet needs to increase the value contained in the field. The SAODV protocol provides two different signature schemes. In the first, each intermediate node will not reply to the request also if it has a route to destination. The second one allows the reply by other nodes, which need to include the original signature of the destination (stored in a cache), signing the fields modified by them. These two approaches are called respectively Single Signature Extension (SSE) and Double

Signature Extension (DSE). Packets generated using these extensions allow each node to verify the messages validity. If the verification fails, the node discards the packet. These are extensions to control packets provided by AODV protocol, containing the information needed to manage SAODV operations. Concerning the hop count field, it has to be modified by each node that forwards the packet, so hash chains are used for this purpose.

A secure protocol based on DSDV, which is a proactive routing protocol, is proposed in [65]. The Secure Efficient Ad-hoc Distance Vector (SEAD) protocol protects the routing updates from malicious nodes by keeping a hash value for each entry in the routing table. The authentication of a route update is achieved through using a hash value computed in a way that does not allow to attackers to advertise a fresher route to a certain destination, because the hash function is one-way.

Authors in [66] proposed an efficient on-demand secure routing protocol, named ARIADNE, providing security against attacks using symmetric cryptography. The point-to-point authentication of routing packets is achieved using a shared key and a message authentication code between the nodes. The authentication of routing messages is guaranteed by the Timed Efficient Stream Loss-tolerant Authentication (TESLA) [67] broadcast authentication protocol. ARIADNE is based on DSR protocol.

A proposal of an on-demand protocol providing secure communications in open environment is presented in [68]. In the Authenticated Routing for Ad-hoc Networks (ARAN) protocol the nodes use session keys that can be exchanged or distributed by a CA. Each node has to authenticate to a trusted certificate server in order to get a certificate. This certificate is used for the authentication to other nodes while exchanging routing information. The existence of a secure path is obtained through storing a route pair, composed of two nodes (previous node and destination node), at each intermediate hop. These fields are concatenated and signed with the source node private key. The route discovery is initialized by the source node, which broadcasts a Route Discovery Packet (RDP). Each node receiving this packet for the first time removes other signatures and signs the packet using its own key, broadcasting again the packet, until the destination is reached. At destination, a reply is generated, signed and sent toward the source.

Security Protocols for Sensor Networks (SPINS) [69] is a suite of two protocols optimized for their use in wireless ad-hoc networks. A modified version of TESLA protocol [67] is introduced for secure broadcast, which supports symmetric cryptographic techniques for authentication, therefore it is more suitable for ad-hoc networks, because the cost of generation and verification of symmetric keys is much less than for asymmetric keys. Point-to-point communication is provided by Secure Network Encryption Protocol (SNEP),

which relies on a counter, shared among the sender and the receiver, in order to ensure security and protect the content of the message.

The Unobservable Secure On-Demand Routing (USOR) scheme is proposed in [70]. It provides a combination of group signature and ID-based encryption for route discovery, in order to protect the privacy of the nodes participating in the wireless ad-hoc network from inside and outside attackers. The main characteristic of this proposal is the impossibility of observing both control and data packets by an external agent. The requirements that the proposal satisfy are:

- anonymity: all the nodes participating in the communication are not identifiable outside the network;
- unlinkability: the knowledge about any relation between two distinct messages is impossible;
- unobservability: any meaningful packet cannot be distinguished from other packets to an outside malicious agent.

The group signature scheme consists in a key server that generates a group public key, known to every node, and a private group signature key for each node in the network. This scheme ensures that a signature does not reveal the identity of the signer. The ID-based encryption scheme is based on elliptic curves. The key server chooses a master secret and generates the ID-based private key for each node.

3.3.5 Other approaches

A novel approach for intrusion detection in pervasive environments is proposed by authors in [71]. The detection of malicious nodes is done by comparing the behavior of the node with a predefined normal profile. The proposed security scheme is divided in three different phases:

1. initialization phase;
2. detection phase;
3. isolation phase.

In the first phase the normal profile of users is defined. This profile includes information such as address, use of CPU, memory occupation and others. The profile is represented by a vector V_i, with $V_i[k]$ representing the value of the attribute k for the user i. The value 0

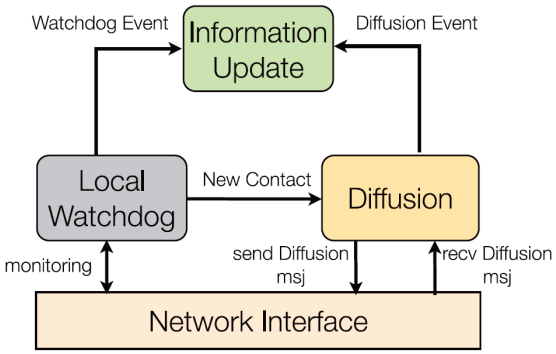


Fig. 3.6 CoCoWa architecture

represents a restriction, while other values represent a privilege. After the initialization, the system switches in detection phase. Information about users behavior is collected, building the current behavior to compare with the normal profile defined in the first phase. If as result of this comparison, the system detects an anomaly, it triggers the isolation phase. The main actions performed in this phase are the communication to other nodes about the intruder, the termination of all connection with the malicious node to remove it from the network, and the tracking of the intruder and the type of attack performed.

The authors in [72] propose a Collaborative Contract-based Watchdog (CoCoWa) to detect selfish nodes in MANETs. Watchdogs are used for network monitoring, they consist on overhearing the packets transmitted in order to detect selfish or malicious behaviors. Each node runs a local watchdog to detect selfish nodes and new contacts. The acquired information is transmitted by the diffusion module, which has also the task of receiving data from other nodes. The system is event-driven, its architecture is shown in Fig. 3.6. The generated events concern positive detections, negative detections and no detection when a node does not have enough information about another node. A threshold is used to avoid the fast spreading of wrong information.

Chapter 4

Energy Consumption Analysis of a Trust Management Scheme in Mobile Ad-hoc Networks

Two of the major issues of MANETs concern the security of exchanged data and the energy consumption. These two properties are in contrast because achieving a better security requires more operations and data exchanges. Introducing a TMS in wireless ad-hoc networks leads to an increase in the energy consumption mainly for the operations involving the wireless interface of the nodes. In the following, the trust model is integrated with a secure encryption-based protocol as additional measure of security against internal attacks. The direct trust evaluation is computed through the analysis of the data gathered by monitoring the transmissions exploiting the promiscuous mode of the wireless interface. Eventually a detailed analysis of the energy consumption in a MANET environment is presented and compared with the results obtained when the TMS is exploited for detecting intruders. As it can be easily foreseen, the main difference in terms of energy consumption is due to the monitoring, which requires additional energy to be executed.

4.1 Trust model

In MANETs, the concept of trust can be defined as the certainty whereby an agent will perform such an action from the subject point of view. The subject is the node requesting the execution of the action (e.g. packet forwarding). The trust value of the agent is computed by controlling its behavior during the time, the observations done through monitoring are

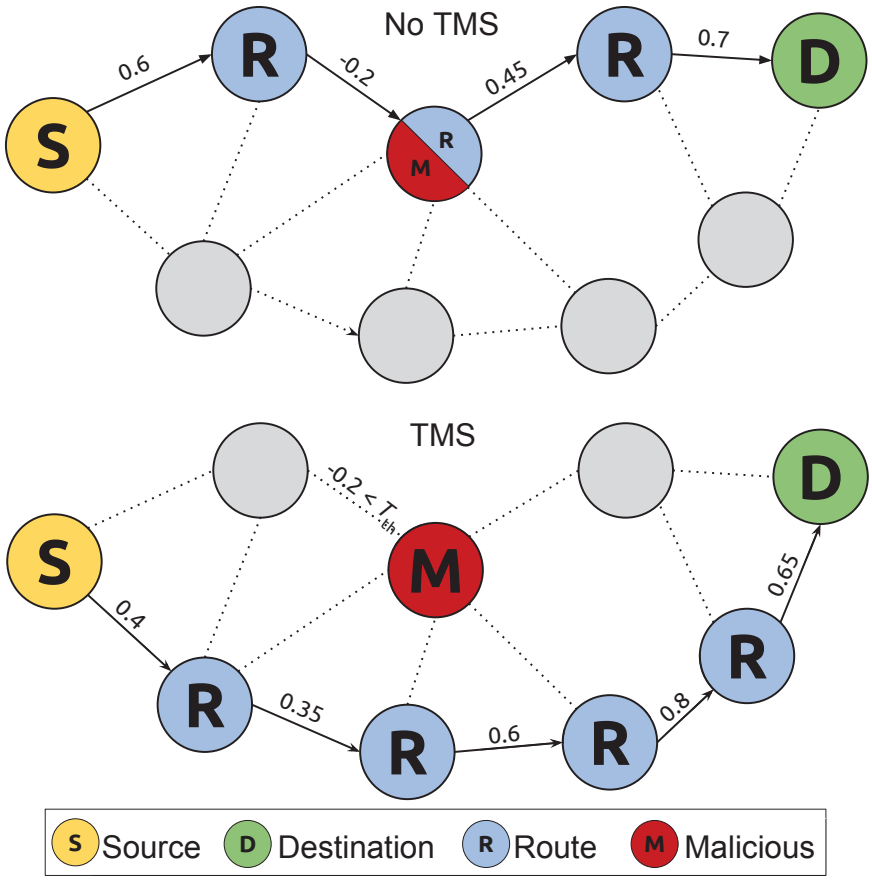


Fig. 4.1 Routing without TMS and with TMS

evaluated to establish if the agent is trustworthy or not. The trust value of an agent is usually included in the ranges [0, +1] or [-1, +1]. Under the subject point of view, the agents with high trust values have an higher probability to perform the action. Usually the mean value of the range in which the trust value is included indicates an uncertainty condition, so the subject has no belief about the agent behavior. A comparison of the expected behavior with and without TMS is shown in the example in Fig. 4.1. Trust value T is reported on each edge composing the chosen route. No TMS scenario allows the inclusion of the malicious node M in the route between the source S and the destination D, because malicious agents cannot be avoided when nodes cannot rely on TMS. The expected behavior of the protocol when a TMS is exploited is shown in the TMS scenario. In this case, malicious agent M is detected and excluded from the route because its trust value T_M dropped below a threshold $T_{\text{th}} = 0$. A trust relationship could be defined as follows:

$$\{subject : agent, action\} \tag{4.1}$$

The trust value T and the probability that the agent will perform the action P are tied to the relationship in Eq. (4.1). The framework of trust modeling proposed in [73] defines a trust value based on the entropy. It allows using interactions and recommendations to compute direct and indirect trust value for the nodes.

4.1.1 Direct trust

The agents behavior could change dynamically, so the trust value T has to also depend on the time. With this aim, a remembering factor ρ is introduced in the formula used for probability computation. ρ value is included in the range [0, 1], the weight of older action observations is higher when its value is near to 1, decreasing the ρ value, the weight decreases too. The remembering factor value has to be chosen depending on the characteristics of the network. The direct trust value of an agent is computed by the subject through direct interactions. The probability P is computed taking into account the observations done as in the following equation:

$$P\{subject : agent, action\} = \frac{1 + \sum_{i=1}^{I} \rho^{t_c - t_i} k_i}{2 + \sum_{i=1}^{I} \rho^{t_c - t_i} n_i} \tag{4.2}$$

The parameters in Eq. (4.2) are defined as follows:

- I: amount of actions observed;
- ρ: remembering factor;
- t_c: current time;
- t_i: time of the observation;
- n_i: observation, with value of 1 for each action observed;
- k_i: successful observation, with value of 1 if the action observed was executed, 0 otherwise;
- P: probability, with value 0.5 when no interactions were observed.

A relationship between ρ and the time t after which an observation n_i has a weight less than $\varepsilon \to 0$ can be defined as follows:

$$\rho_t = \varepsilon^{1/t} \tag{4.3}$$

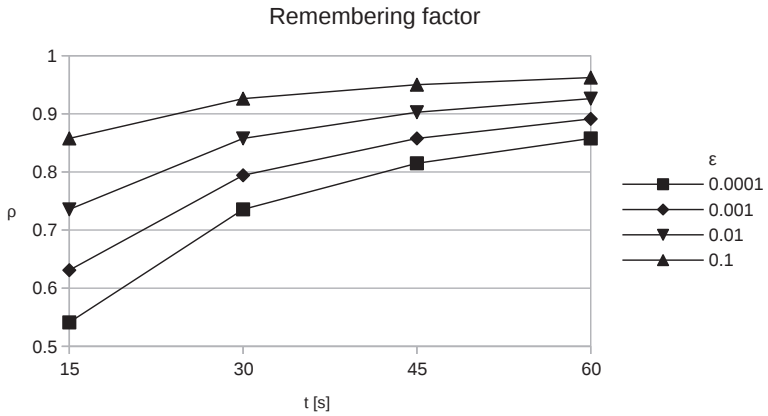


Fig. 4.2 Values of ρ_t depending on t and ε

The trend of remembering factor ρ_t depending on ε value is shown in Fig. 4.2. Having already assigned a value to ρ_t, the time t can be computed as follows:

$$t = \frac{\log \varepsilon}{\log \rho_t} \tag{4.4}$$

The trust value computation is based on the following entropy function:

$$H(P) = -P\log_2(P) - (1-P)\log_2(1-P) \tag{4.5}$$

The entropy is referred to the uncertainty in the information theory. Equation (4.5) is exploited to compute the trust value of an agent. It can assume values among -1 and $+1$, for $p = 0.5$ it has the value of 0 (highest uncertainty about the action execution).

$$T = \begin{cases} 1 - H(P), & \text{for} \quad 0.5 \leq P \leq 1 \\ H(P) - 1, & \text{for} \quad 0 \leq P < 0.5 \end{cases} \tag{4.6}$$

4.1.2 Indirect trust

A recommendation system can be used to evaluate the trustworthiness of an agent before requesting an action if its current trust value can not be computed through direct interactions (i.e. no recent interactions between the subject and the agent). The *recommendation* action needs an evaluation in terms of trust value too. The evaluation of the trust of an agent through recommendations has to satisfy three properties:

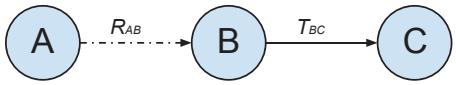


Fig. 4.3 Trust chain

1. concatenation propagation of trust does not increase trust (i.e. the trust value of a node should not be higher than the trust of "recommendation" action of the recommender agent). An graphical representation of this property is shown in Fig. 4.3);

$$|T_{AC}| \leq \min\left(|R_{AB}|, |T_{BC}|\right) \tag{4.7}$$

2. multipath propagation of trust does not reduce trust (i.e. when receiving more than one recommendation, the trust value of an agent should not be lower than the value obtained upon the reception of only one of the recommendations). This property is illustrated in Fig. 4.4a);

$$T_{A_2C_2} \geq T_{A_1C_1} \geq 0, \quad \text{for } R_1 > 0, T_2 \geq 0 \tag{4.8}$$

$$T_{A_2C_2} \leq T_{A_1C_1} \leq 0, \quad \text{for } R_1 > 0, T_2 < 0 \tag{4.9}$$

3. multiple recommendations from a single source (represented in Fig. 4.4b) should not be higher than recommendations obtained from independent sources.

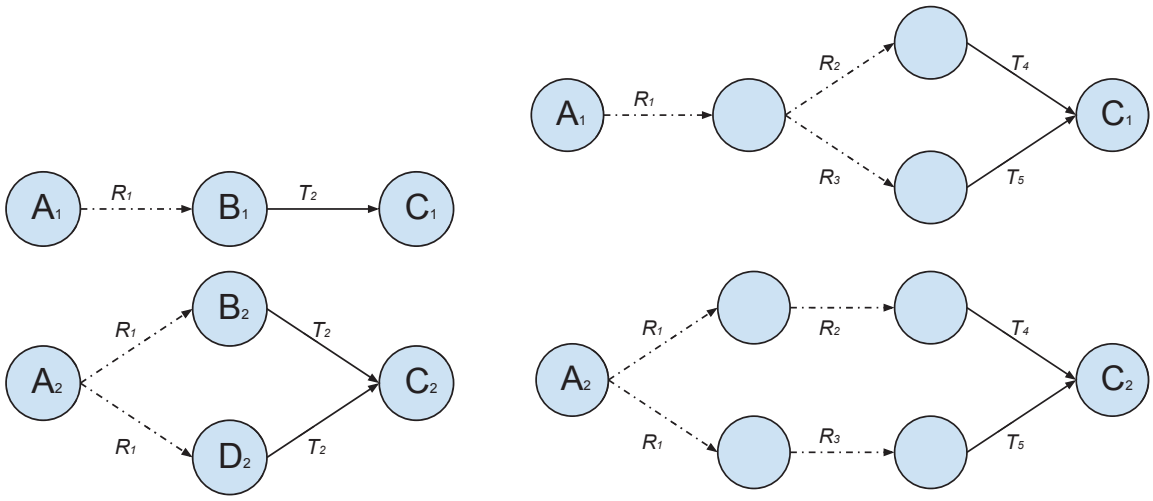
$$T_{A_2C_2} \geq T_{A_1C_1} \geq 0, \quad \text{if } T_{A_1C_1} \geq 0 \tag{4.10}$$

$$T_{A_2C_2} \leq T_{A_1C_1} \leq 0, \quad \text{if } T_{A_1C_1} < 0 \tag{4.11}$$

The adopted entropy-based model respects the property defined by Eq. (4.7) regarding trust chains. The computation of the trust value for an agent C for a subject A through a recommendation received from a node B is the following:

$$T_{AC} = R_{AB}T_{BC} \tag{4.12}$$

The term R_{AB} represents the trust value for the recommendation action of the agent B for the subject A. If the subject A receives more than one recommendation (e.g. from B and D), the



(a) Multipath propagation of trust (b) Multiple recommendations from the same source

Fig. 4.4 Recommendations management

following formula is used:

$$T_{AC} = \omega_1(R_{AB}T_{BC}) + \omega_2(R_{AD}T_{DC}) \tag{4.13}$$

The coefficients of Eq. (4.13) are calculated as follows:

$$\omega_1 = \frac{R_{AB}}{(R_{AB} + R_{AD})} \tag{4.14}$$

$$\omega_2 = \frac{R_{AD}}{(R_{AB} + R_{AD})} \tag{4.15}$$

Equation (4.13) satisfies the properties 2 and 3 defined for trust propagation.

4.2 Energy model

The most power consuming operation performed by a node in wireless ad-hoc networks is the wireless transmission. The used energy model is linear, the fixed cost b represents the cost for accessing the channel, and the incremental cost m depends on the packet size [74]. The values of these parameters were empirically obtained. The reasons behind the choice of this model against more recent ones (e.g. a model based only on radio state as in [75]) concern the costs about the packets reception in promiscuous mode, which are explicitly

defined in the chosen model.

$$\text{Cost} = m \times \text{size} + b \tag{4.16}$$

The total cost of a packet in the network is the sum of the transmission cost by the sender and all the costs of the potential receivers, which include nodes in transmission range of the sender and the destination. The wireless interface has four different states: receiving, transmission, idle and sleep. The last one is not used in the ad-hoc networks because it does not allow receiving or transmitting data. Therefore, the nodes not involved in transferring data stay in the idle state. For a broadcast transmission, the cost includes listening to the channel by the sender as fixed cost, while transmitting the packet and receiving it for all the nodes in the sender wireless range is a variable cost, as shown in the following equation:

$$\text{Cost}_{\text{BC}} = m_{\text{Tx}} \times \text{size} + b_{\text{Tx}} + \sum_{n \in S} (m_{\text{Rx}} \times \text{size} + b_{\text{Rx}}) \tag{4.17}$$

Set S refers to all the nodes included in the transmission range of the sender node. Point-to-point transmission has a cost that includes Request to Send (RTS), Clear to Send (CTS) and Acknowledgment (ACK) messages used in the 802.11 MAC protocol. The cost to transmit or receive one of them indifferently is represented by $b_{\text{Tx}_{\text{ctl}}}$ and $b_{\text{Rx}_{\text{ctl}}}$. The cost for the destination is similar, with the transmission and the reception phases inverted respect to the sender. These costs are respectively shown in Eq. (4.18) and Eq. (4.19).

$$\text{Cost}_{\text{src}} = b_{\text{Tx}_{\text{ctl}}} + b_{\text{Rx}_{\text{ctl}}} + m_{\text{Tx}} \times \text{size} + b_{\text{Tx}} + b_{\text{Rx}_{\text{ctl}}} \tag{4.18}$$

$$\text{Cost}_{\text{dst}} = b_{\text{Rx}_{\text{ctl}}} + b_{\text{Tx}_{\text{ctl}}} + m_{\text{Rx}} \times \text{size} + b_{\text{Rx}} + b_{\text{Tx}_{\text{ctl}}} \tag{4.19}$$

The nodes in the range of the source or the destination discard the packet if the transmission is point-to-point and they are not in promiscuous mode. Some MAC implementations allow entering in an energy-saving mode state when a transmission directed to another node is detected. The cost at non-destination nodes is the following:

$$\text{Cost}_{\text{no\_dst}} = \tag{4.20}$$
$$\sum_{n \in S} b_{\text{discard}_{\text{ctl}}} + \sum_{n \in D} b_{\text{discard}_{\text{ctl}}} + \sum_{n \in S} (m_{\text{discard}} \times \text{size} + b_{\text{discard}}) + \sum_{n \in D} b_{\text{discard}_{\text{ctl}}}$$

All the nodes in the destination range are in the D set. The cost for discarding control packets is represented by $b_{\text{discard}_{\text{ctl}}}$. If a node works in promiscuous mode (e.g. when it needs to sense if a node forwards a packet for computation of the trust value), the cost calculation is the same as in Eq. (4.20), except for m_{discard} and b_{discard}, which are changed respectively with $m_{\text{Rx}_{\text{prom}}}$

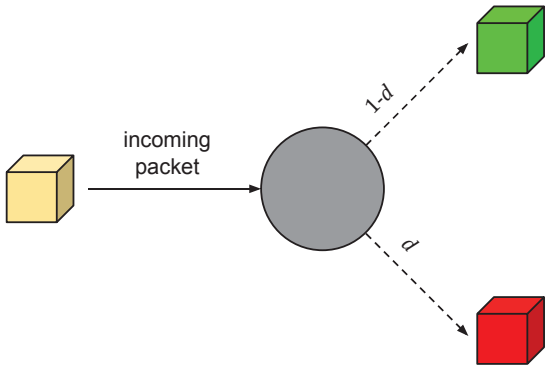


Fig. 4.5 Attacker with random drop probability d

and $b_{\mathrm{Rx}_{\mathrm{prom}}}$. Cryptographic algorithms involve energy consumption due to the computational time they need to compute hash digests and to generate and verify signatures. The adopted energy model for these operations has a fixed cost for each signature or verification, while the cost of applying the hash function is incremental. These costs depend on the algorithms used [76].

$$\mathrm{Cost}_{\mathrm{sign|verify}} = b_{\mathrm{sign|verify}} \tag{4.21}$$

$$\mathrm{Cost}_{\mathrm{hash}} = m_{\mathrm{hash}} \times \mathrm{size} \tag{4.22}$$

4.3 Attacker model

The attacker model is important to analyze the response of an IDS against a specific attack. A malicious node performing a black hole attack tries to redirect most of the traffic in the network through it, then dropping all the data packets that it receives [27]. The gray hole attack is very similar to the previous one. It does not drop all packets, but just a part of them, based on the data contained in the packet or randomly with a certain probability [30]. Malicious nodes in the network are based on this type of attack, dropping some of the data packets received using a certain logic (e.g. random, based on the packet originator, as a function of the residual energy), but fairly participating to route discovery and maintenance operations. They change their behavior depending on various parameters, therefore the detection of this attack is difficult. The model of a malicious node that drops a percentage of packets defined as d is shown in Fig. 4.5. Each time it receives a packet, the choice of dropping or forwarding the data packet is taken by generating a random number. The detection of malicious nodes with a low d value is harder. Algorithm 1 represents the way in

which a malicious node chooses to forward a packet or not. Only data packets are managed by the algorithm, because a malicious agent can be a threat to the network only if packets go through it, and using a secure protocol prevents modification, impersonation and fabrication attacks [64], which can be used by an agent to be part of more routes. Concerning only data packets, the choice of d defines which type of attack is performed. The lowest value of d can be 0, meaning that the node is fair and no packets are dropped; when d is set at its highest possible value ($d = 1$), all data packet are dropped by malicious agent. For all the values of d included between 0 and 1, a percentage of data packets is randomly dropped with probability d.

Algorithm 1 Malicious node algorithm

```
for packet = new data packet received do
    rnd = random generated number
    if rnd ≤ d then
        DROP packet
    else
        FORWARD packet
    end if
end for
```

4.4 Proposal: Secure and Trusted AODV

Introducing a trust management procedure in a protocol that provides already encryption procedures to authenticate messages and hash functions to protect mutable fields could increase the kind of attacks that the network can detect. The proposed Secure and Trusted AODV (STAODV) protocol [28] concerns an extension of SAODV protocol, exploiting a TMS to protect the network against malicious nodes that participate correctly to the route establishment phase, but then they behave maliciously by executing a packet dropping attack, extending the protection to the attacks made ineffective by digital signatures and hash chains in SAODV.

4.4.1 Trust Management Scheme

To improve the security of the network against malicious and selfish nodes, the TMS defined in Section 4.1 is introduced, so the packets will follow routes composed by trusted nodes, avoiding the malicious ones. The protocol manages the RREQ packets as SAODV protocol

already do, broadcasting only the first one and discarding others with the same sequence number, but updating routing table adding the new next hop to the originator. When a node with a valid route to the destination is discovered, or the packet reaches the destination, a RREP packet will be sent to the node that forwarded the request. The requests with the same sequence number are managed by generating a fixed maximum number of replies at the destination. The RREP packets follow the path to the originator, they are forwarded in unicast.

Recommendation packets

In the proposal, two new packet typologies are introduced to manage indirect trust, allowing the nodes to send and receive recommendations:

- Trust Recommendation Request (TRREQ);
- Trust Recommendation Reply (TRREP).

TRREQ packet contains the request originator and a list of the requests about the agents of which the originator needs to know if they are reliable or not. TRREP packet contains the request originator, the recommender, who generates the reply, and a list of pairs containing the agent and its trust value T. The signature extension exploited by SAODV protocol to secure control packets is added in a similar way to recommendation packets too.

Trust table

The nodes need to store the trust values about the agents. Concurrently with the routing table, a trust table is introduced. Each entry of the table contains the agent and the parameters that allow computing the trust value and updating it when needed. The table is updated periodically, so the recommendations and the direct observations are stored in buffers until the update. If a node has a trust value about forwarding packet less than 0, it is included in a blacklist, and it is removed from all the routes stored. If a route has only the blacklisted node as next hop, the route is invalidated, and a new route discovery process is launched when the node needs to reach that destination. For $\rho < 1$, the trust becomes 0 when enough time elapses, and the blacklisted node is removed from the blacklist.

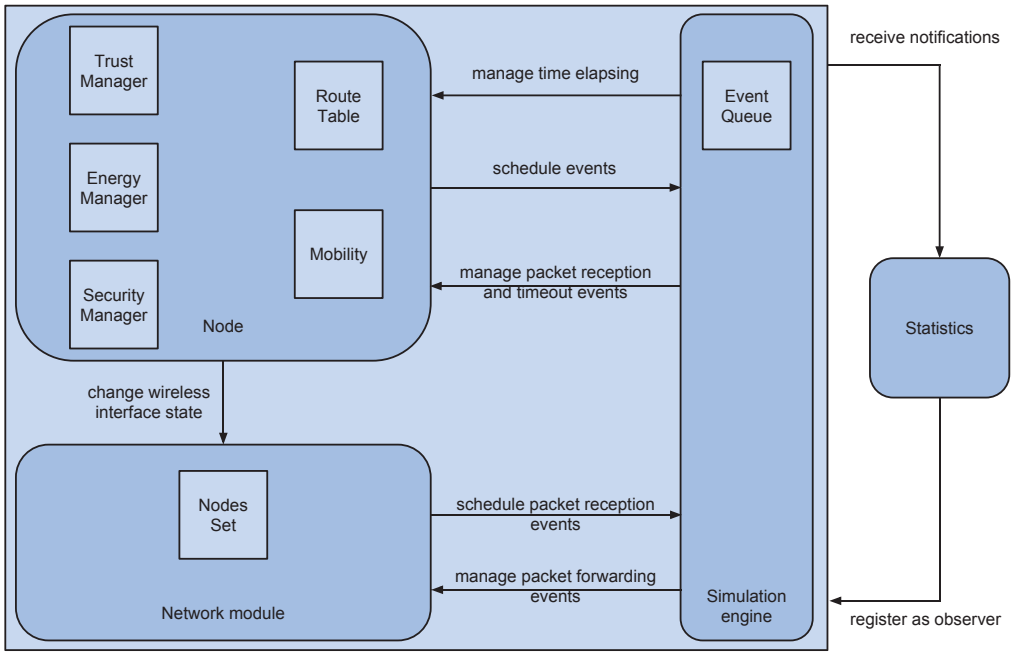


Fig. 4.6 Simulator architecture

Table 4.1 Simulation Parameters

Parameter	Value
Area	1000×1000 m^2
Duration	10 minutes
Transitory time	1 minute
Nodes	50
Transmission range	250 m
Transmission rate	4 packets/s
Data packet size	128 bytes

4.5 STAODV performance evaluation

STAODV protocol is compared with the SAODV protocol under the performance and the energy consumption point of view. The simulator was developed following the architecture shown in Fig. 4.6. The parameters used to run simulation campaigns are shown in Table 4.1. The remembering factor in Eq. (4.2) is set as $\rho = 0.9$ for the action *transmission*, using the second as time unit in the equation, so the trust value resets to 0 after 60 seconds, "rehabilitating" a node in the blacklist. The action *recommendation* has $\rho = 1$, meaning that the recommendation trust value does not expire.

The movement model adopted by the nodes in the simulation is the Random Way-Point (RWP) model [77]. Each node goes from a random point in the space to another following a

Table 4.2 Energy Model Parameters

Parameter	Value
m_{Tx}	1.89 mJ/byte
b_{Tx}	246 mJ
m_{Rx}	0.494 mJ/byte
b_{Rx}	56.1 mJ
m_{discard}	−0.49 mJ/byte

Parameter	Value
b_{discard}	97.2 mJ
$m_{\text{Rx}_{\text{prom}}}$	0.388 mJ/byte
$b_{\text{Rx}_{\text{prom}}}$	136 mJ
$b_{\text{Tx}_{\text{ctl}}}$	120 mJ
$m_{\text{Rx}_{\text{ctl}}}$	−0.49 mJ

Parameter	Value
b_{sign}	546.5 mJ
b_{verify}	15.97 mJ
m_{hash}	0.76 μJ/byte

linear path, with a speed calculated randomly between 0 and the maximum allowed speed. When the destination is reached, a new random destination is randomly chosen. There is no pause time between the reaching of the destination and the starting to the new one. For each combination of parameters, we have run 3 simulations, using the resulting average values to plot the graphs. Regarding the consumption analysis, the values used for the energy model are shown in Table 4.2, as obtained empirically in [74, 76]. Malicious nodes in the network perform a packet dropping attack with different dropping probabilities, because the SAODV protocol is already resistant to black hole and gray hole attacks.

To evaluate the reliability of the recommendation system, nodes disseminating false recommendations were included in the network. Performing this kind of attack has two main purposes: distrust a fair node, or trust a malicious node. The protocol has to detect the nodes running that kind of attack, so their recommendations can be excluded from the indirect trust computation process.

The simulations were run to compare the performance of the two protocols with and without malicious nodes in the network. The following graphs will show the improvement that could be achieved using a trust management scheme in the mobile ad-hoc networks under different points of view.

4.5.1 Packet Delivery Ratio

The evaluation of the Packet Delivery Ratio (PDR) is one of the most important characteristic of a protocol. By analyzing it, we can understand if a protocol is susceptible to a threat or

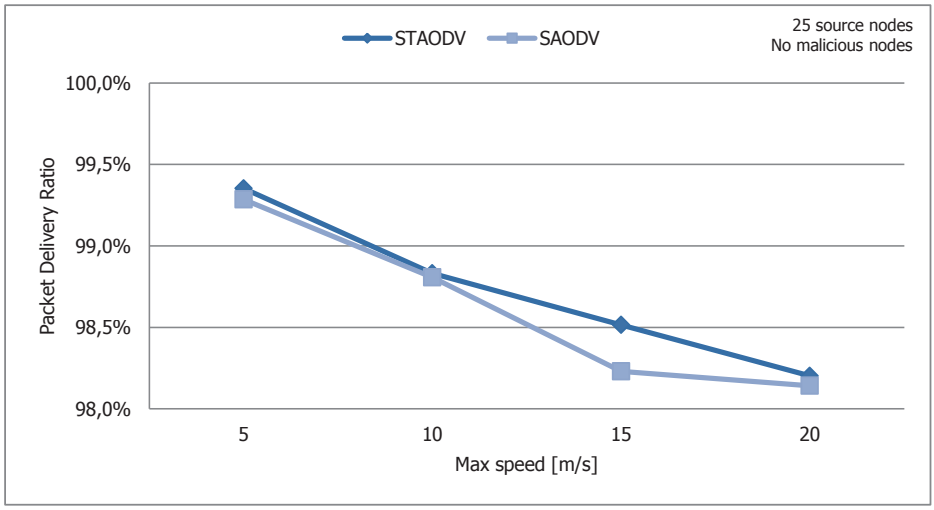


Fig. 4.7 PDR without malicious nodes

not. To compute this ratio for the entire network, the following equation was used:

$$\text{PDR} = \frac{\#packets\_received}{\#packets\_sent} \tag{4.23}$$

In MANETs, the PDR decreases when the nodes speed increases, even if no malicious nodes are in the network. The cause is the breakage of links that incurs frequently when the nodes move faster. The performance of the two protocols in a network without malicious nodes are very similar, as shown in Fig. 4.7. PDR is always above 98% until a maximum speed of 20 m/s. There is a small decrement in both protocols when the speed increases, due to the reasons explained before.

The performance in terms of PDR when the network is threatened by malicious nodes that drop the 75% of received data packets is shown in Fig. 4.8. The ratio decreases in both protocols when the number of malicious nodes increases. Comparing the two protocols, STAODV maintains a ratio of almost 90% even when the 30% of the nodes in the network execute this attack, while the SAODV achieves a PDR lower than the 75%. When 5 nodes are malicious, the PDR of SAODV protocol is near the 90%, but it is less than the result obtained with the trust management scheme. With almost 1/3 of the nodes misbehaving, the protocol can deliver less than 3 packets each 4 packets sent, which is an awful result.

The performance of the protocols when malicious nodes of the network drop the 100% of data packets is shown in Fig. 4.9. The difference between the two protocols is higher respect to the previous attack. The STAODV protocol continues to keep a PDR higher than the 90%, while the PDR of the SAODV protocol decreases under the 65% when malicious nodes are

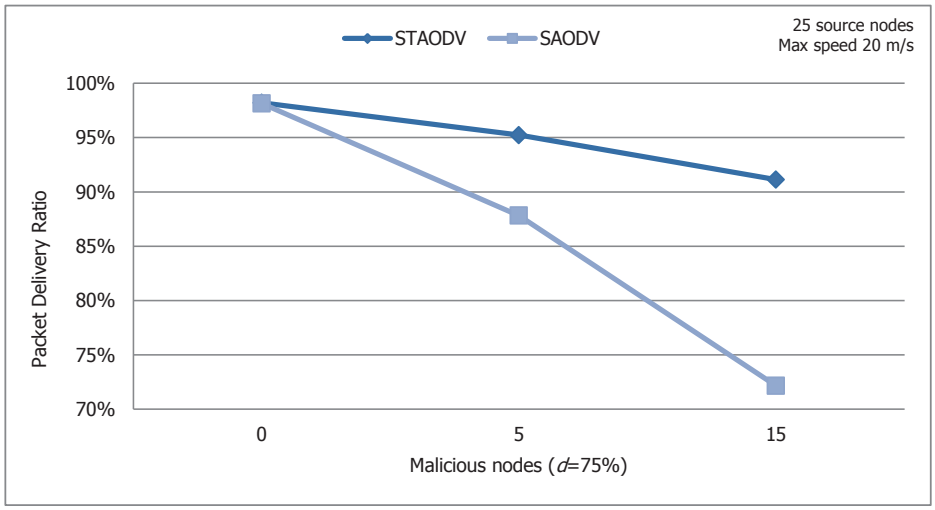


Fig. 4.8 PDR with malicious nodes ($d = 75\%$)

the 30% of the entire network. Therefore, SAODV is able to deliver only the packets that follow paths composed of few nodes, which have a minor probability to include a malicious node.

The comparison of the effects that the dropping percentage applied by malicious nodes have on the PDR for the STAODV protocol is shown in Fig. 4.10. The protocol reacts well against both the threats, reaching a better result in detecting and avoiding malicious nodes dropping all the data packets when they are the 10% of the total amount of nodes that compose the network. When the number of malicious nodes increases, the STAODV achieves almost the same PDR for both the attacks.

4.5.2 False recommendations detection

To evaluate the reliability of the recommendation system, nodes disseminating false recommendations were included in the network. Performing this kind of attack has two main purposes: distrust a fair node, or trust a malicious node. The protocol has to detect the nodes running that kind of attack, so their recommendations can be excluded from the indirect trust computation process. A TMS should be able to distinguish the fair recommendations from the false ones. For this reason, the nodes maintain a trust value about recommendations for each node. In Table 4.3 the average trust values of the nodes in the network is shown. The cells highlighted in red correspond to the values concerning the malicious nodes. The analysis of these values shows that the protocol detects misbehaving nodes by spreading false information, because their average trust value on recommendations in the network is notably under zero. There are

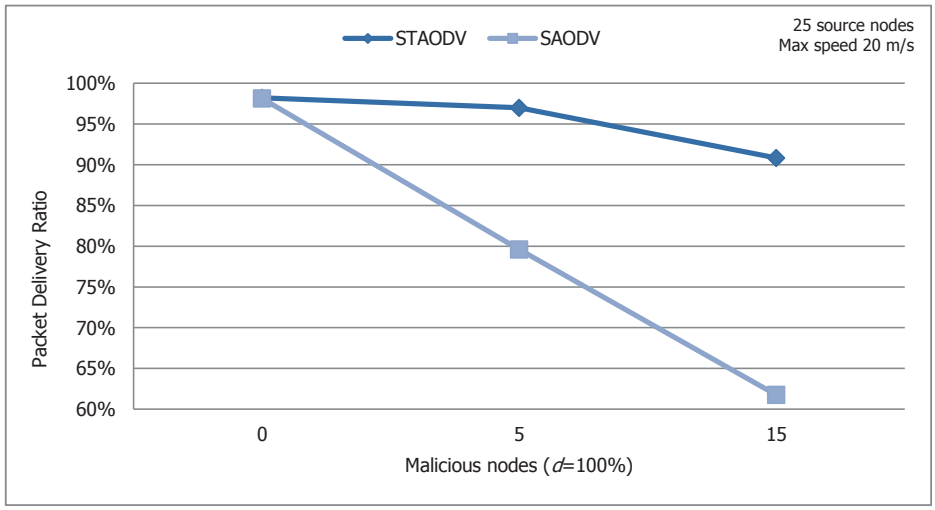


Fig. 4.9 PDR with malicious nodes ($d = 100\%$)

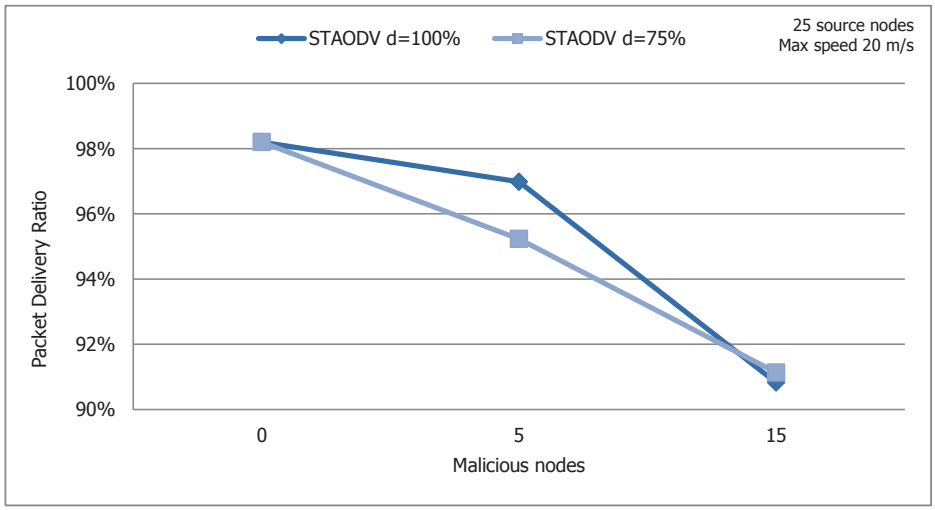


Fig. 4.10 PDR against malicious nodes with different d values

Table 4.3 Average Trust Values

5 false recommenders					10 false recommenders				
0.056	0.066	0.07	0.03	*-0.018*	0.047	0.046	0.044	*-0.004*	***-0.308***
0.086	0.059	0.012	0.064	0.019	0.004	0.11	0.053	0.02	***-0.276***
0.08	*-0.036*	0.142	*-0.01*	0.083	0.054	0.009	0.037	0.073	***-0.305***
0.062	*-0.027*	0.043	0.025	0.012	0.096	*-0.014*	0.07	0.027	***-0.34***
0.023	*-0.014*	0.025	0.033	0.008	0.078	0.08	0.095	0.087	***-0.397***
0.009	0.006	0.068	0.016	***-0.32***	0.08	0.037	0.054	0.036	***-0.323***
0.065	0.034	0.045	0.037	***-0.325***	0.063	0.05	0.027	0.057	***-0.35***
0.1	*-0.018*	0.083	0.088	***-0.303***	0.066	0.054	0.015	0.052	***-0.35***
0.08	0.084	0.075	0.05	***-0.297***	0.032	0.094	*-0.001*	0.016	***-0.33***
0.103	0	0.042	0.045	***-0.314***	0.036	0.106	0.055	0.034	***-0.273***

also other negative values, but near to zero, for some fair nodes. The reason can be found in the link breakages that sometimes occur in a wireless ad-hoc network, so a node cannot forward a packet to its path and a fair recommendation could be evaluated as malicious. This situation occurs rarely with respect to the malicious recommendations, so the values are not so distant from zero, and a node can raise its trust value over the zero if its opinion is needed to other indirect trust evaluations. To avoid the distrust of a fair node, the protocol provides a threshold before considering a node as malicious. A recommendation is evaluated exact if the difference between it and the computed direct trust value is in a range of ± 0.25.

4.5.3 Erroneous detections

When a TMS is used, the detections have to be correct, because distrusting a fair node increases unnecessarily the length of the paths, increasing the End to End (E2E) delay too. However the links between the nodes in a acMANET often break, therefore completely avoiding erroneous detections is impossible. After enough time elapses, STAODV protocol allows the participation in the network to any node also if it was previously detected as malicious.

The percentage of the erroneous detections increases with the maximum moving speed of the nodes, as shown in Fig. 4.11. When the nodes move faster, the amount of broken connections among the nodes increases, so a node could evaluate the behavior of the agent as malicious. However, the percentage of fair detections remains higher than 95% also when the maximum speed is 20 m/s, which means that there is an erroneous detection less than every 20 detections. For lower speeds, the percentage of fair detections is higher, because the links between the nodes are more stable.

Fig. 4.12 shows the detections related to the amount of the source nodes in the network. The erroneous detections percentage increases proportionally, because there are more packets in the network. Therefore, if a node cannot forward a packet before the detection time of the sender node expires, it will be judged as malicious even if it will forward the packet in a later time, because the node will not monitor the agent anymore. The erroneous detections are independent from the amount of the malicious nodes inside the network, as we can see in Fig. 4.13, so the percentage of fair detection increases with the amount of malicious nodes because more malicious behaviors are detected.

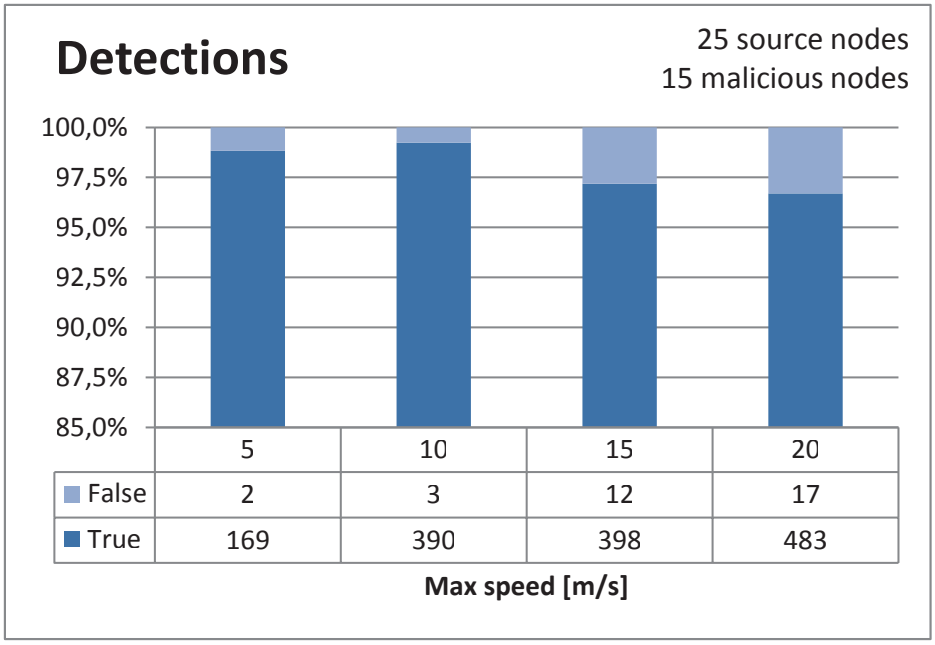


Fig. 4.11 Erroneous detections with respect to the maximum speed

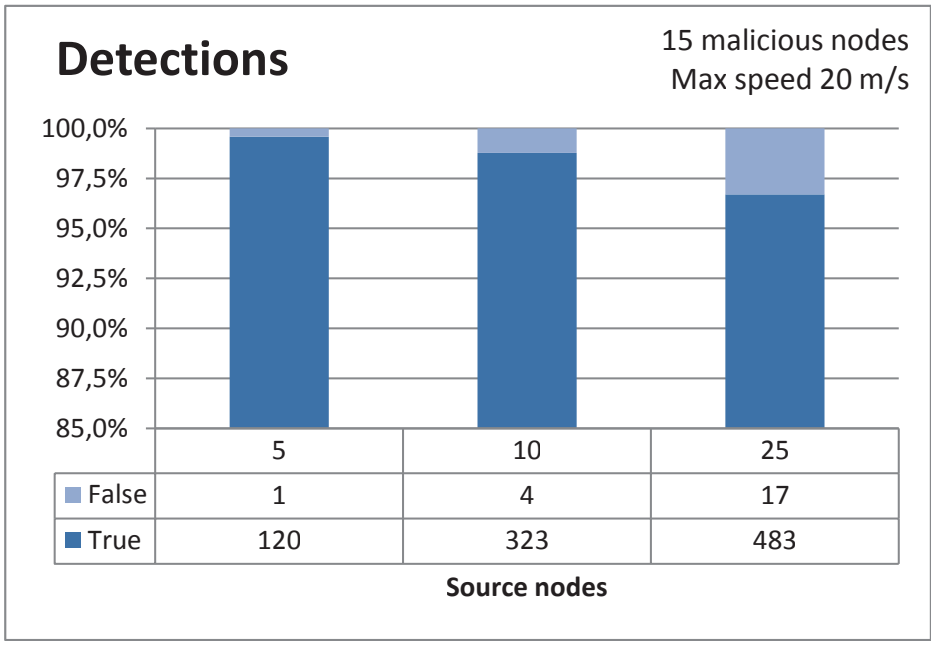


Fig. 4.12 Erroneous detections with respect to the amount of source nodes

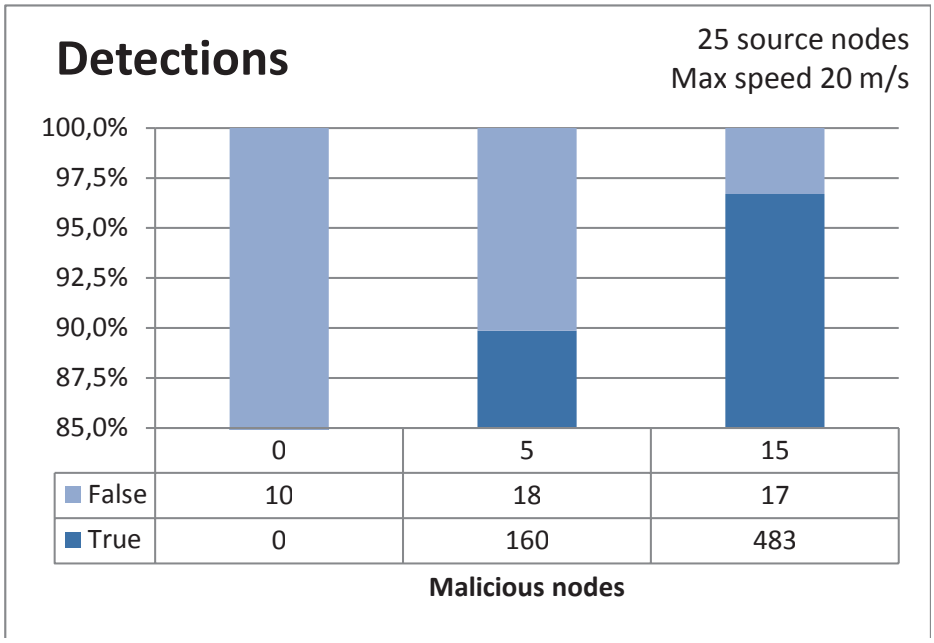


Fig. 4.13 Erroneous detections with respect to the amount of malicious nodes

4.5.4 Energy consumption analysis

The nodes in MANETs are powered by batteries, so their energy availability is limited. A protocol for wireless ad-hoc networks needs to take in account this limitation, providing a way to route data without exceeding in energy needed to run the protocol. The most energy consuming operations required by a MANET routing protocol are the wireless communication and the cryptographic operations (e.g. signatures and verifications). With this analysis we study the consumption of SAODV and STAODV protocols, trying to analyze the impact of the introduction of the TMS in an ad-hoc routing protocol.

The energy consumption of the wireless transmission when only fair nodes are in the network is shown in Fig. 4.14. STAODV protocol requires more energy than the SAODV. When the maximum speed increases, the consumption increases in both protocols too. The main difference in terms of power consumption is introduced by the promiscuous mode used in STAODV to let the nodes sensing when their neighbors forward a packet correctly.

Fig. 4.15 shows that the difference between the two protocols is higher when the network is under attack. SAODV protocol energy consumption seems to not increment when the maximum speed increases, because just the packets that follow a short path can reach their destinations, otherwise the probability that a malicious node is included in the path is higher, and the packet will be dropped. This occurrence leads to an energy saving due to the nodes that do not receive and forward the packet. The energy consumption of the STAODV protocol

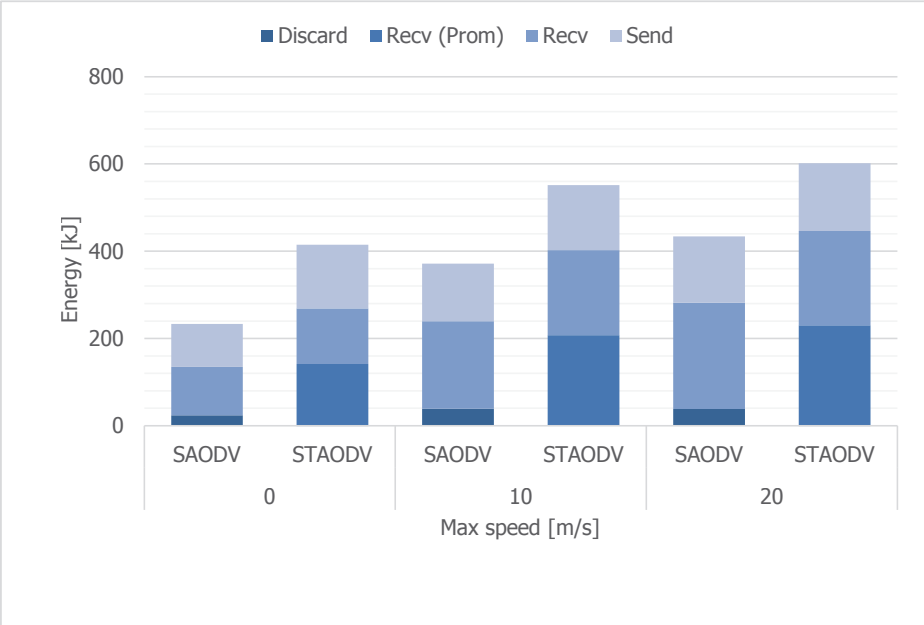


Fig. 4.14 Energy consumption of the wireless transmission without malicious nodes in the network

increases at higher speeds, because with more link breakages and more distrusted nodes, the nodes have to start more route discovery processes, with routes composed of more nodes.

The cryptographic operations consume energy because computing a digital signature, verifying it and applying a hash function are expensive operations in terms of CPU usage. The STAODV protocol requires the application of a signature on more packets than SAODV because it introduces two new control packets to manage recommendations. The energy consumption in the network with only fair nodes is shown in Fig. 4.16. The consumption of the STAODV protocol is higher when nodes are moving, whilst it is almost the same of the SAODV when the nodes are stationary. Moreover, there is an increment in energy consumption in both protocols when the nodes move faster, but it is higher when TMS is used. In the SAODV protocol, the consumption increases due to the route discovery phases, which are triggered more often. It occurs in STAODV protocol too, but it also requires that the nodes ask for recommendations if they have to send a packet to a new neighbor, and it usually occurs when a new route is discovered.

Fig. 4.17 shows that the presence of malicious nodes in the network involves a higher energy consumption for STAODV. Using the TMS, the routes are invalidated when a malicious node is part of them, so the STAODV launches more route discovery processes, increasing the amount of signatures, verifications and hashes needed. Despite a packet is signed only when generated and verified each time a node receives it, the consumption due to

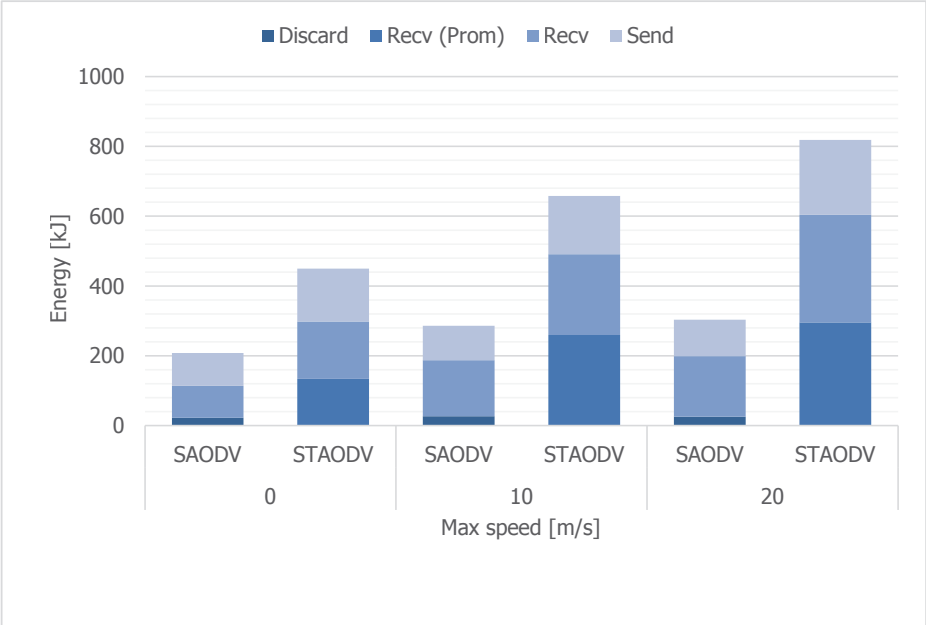


Fig. 4.15 Energy consumption of the wireless transmission with malicious nodes in the network

signing operations is higher, because the generation of a signature using the RSA algorithm requires more than 30 times the energy needed to verify it [76]. All the previously analyzed graphs do not show the energy consumption due to the execution of the hash function, because it is much lower than other cryptographic operations.

The percentage of consumption due to wireless transmission and cryptography respect to total energy consumption seems not to depend on the existence of malicious nodes in the network, as can be seen in Fig. 4.18. When nodes are static, cryptography consumption is almost the 20% of the overall consumption in nearly all the scenarios. The wireless transmission has a higher impact on the overall energy consumption when the nodes in the network move, reaching almost the 90%.

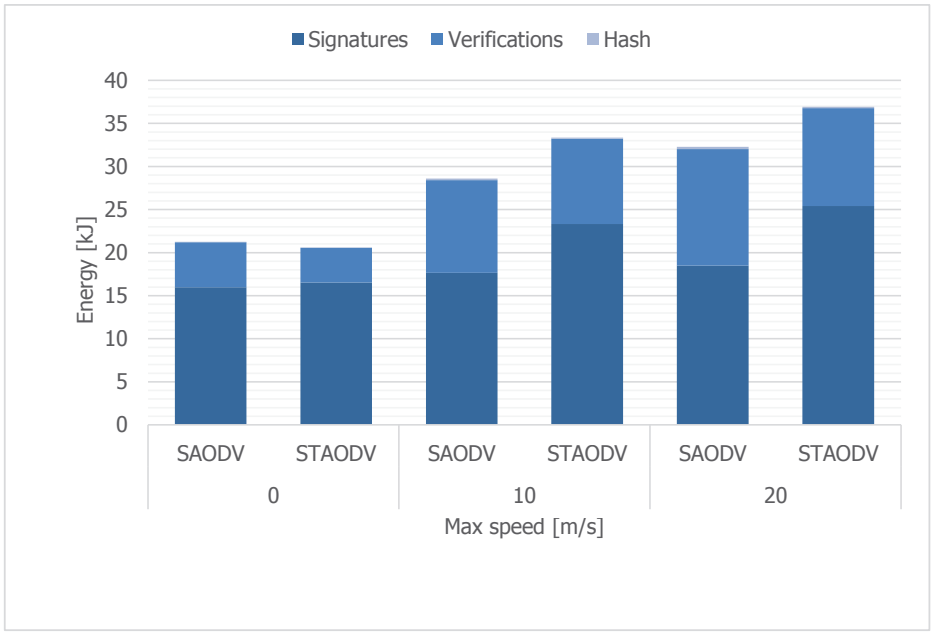


Fig. 4.16 Energy consumption due to cryptographic operations without malicious nodes in the network

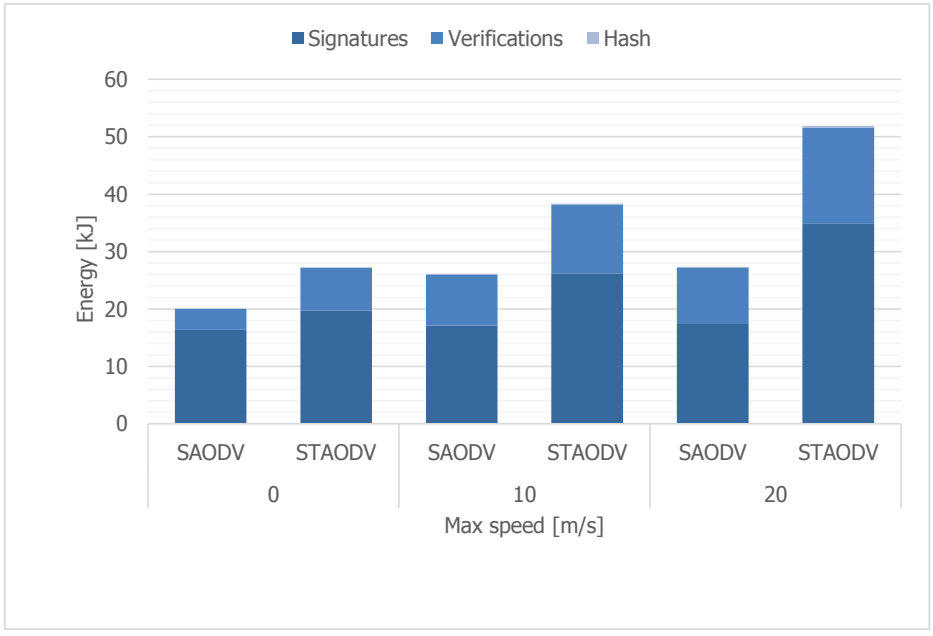


Fig. 4.17 Energy consumption due to cryptographic operations with malicious nodes in the network

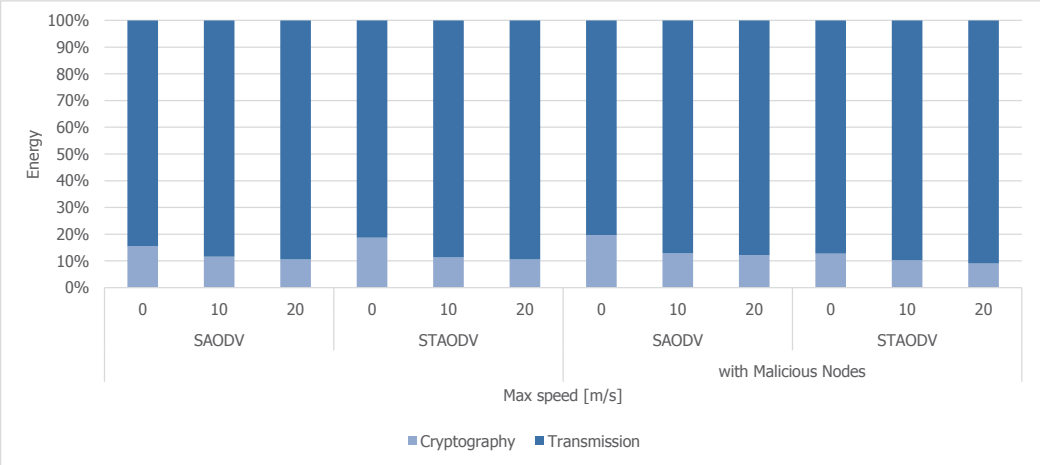


Fig. 4.18 Comparison of energy consumption due to wireless transmission and cryptographic operations

Chapter 5

Distributed Intrusion Detection based on Trust Management, Time Division Monitoring and Link Duration Estimation

In wireless ad-hoc networks, IDSs detect malicious and selfish nodes usually exploiting the promiscuous mode of the wireless interface to overhear the traffic, therefore detecting misbehaviors. This operation leads to an increase in energy consumption, which is a severe issue in networks composed of resource-constrained nodes. Distributing the monitoring through many trusted nodes could help in saving energy while maintaining the system protected against threats. In the following, a distributed trust-based monitoring solution able to face both malicious and selfish nodes is illustrated [78]. It combines trust evaluation and link duration in order that a node can select the most trusted and stable nodes among ts neighbors. Afterwards, nodes will mutually synchronize in a distributed way, dividing the time in which they perform monitoring operations between all them. Achieving this goal is made possible through some additional fields added to messages used by protocols to keep alive the connection between neighbor nodes. With this technique the energy consumption is significantly reduced, depending on the amount of trusted neighbors of each node, because reducing the time dedicated to monitoring will help in extending nodes lifetime.

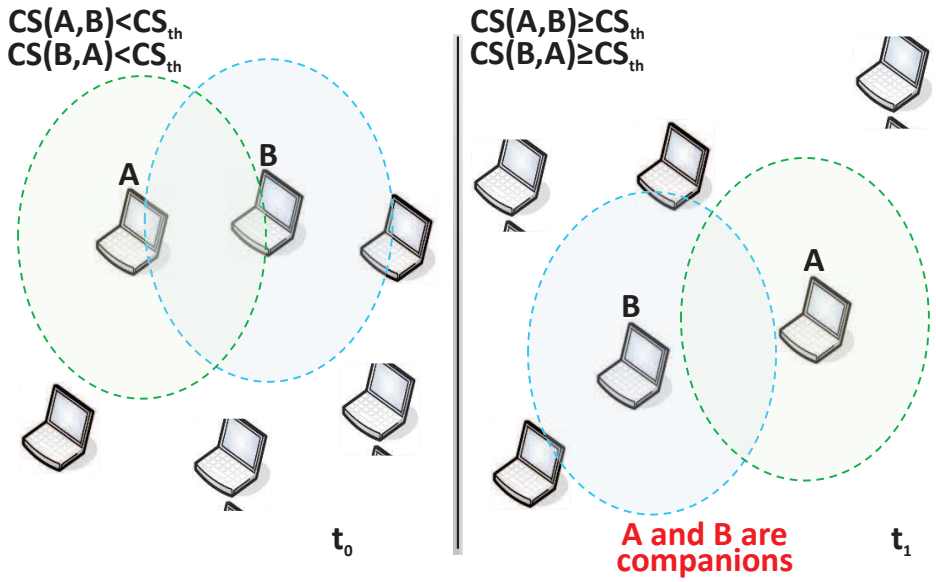


Fig. 5.1 Companions with high reciprocal trust value and similar mobility patterns

5.1 Trust and link duration computation

A node A is a Companion Node (CN) of B if the node B has a high trust value from the node A perspective, and the communication medium is mostly stable between them. The concept of Companion Score (CS) is introduced, representing a score dependent on the trust value and the link stability of a node B from the node A point of view. It is computed through the following equation:

$$\mathrm{CS}(A,B) = (\alpha \cdot T_{AB}) + [(1-\alpha) \cdot \mathrm{LSI}(A,B)] \tag{5.1}$$

Where $0.5 < \alpha < 1$ in order to give more importance to the nodes trustworthiness. The trust value T_{AB} is computed through direct interactions, as shown in Section 4.1.1. Link Stability Index (LSI) represents an estimation of the stability of a link, which computation is described in the following of this Chapter.

A Companion Table (CT) is exploited to store a reference of CNs. Therefore, node B is included in the CT only if its CS is higher than a predefined threshold $\mathrm{CS_{th}}$, as illustrated in Fig. 5.1. When $\mathrm{CS}(A,B)$ value goes below the same threshold, the node is removed from CT.

5.1.1 Link Stability Index computation

The LSI is derived from the work in [79], where the authors proposed a link stability-aware metric to conform to the scalability properties required by wireless ad-hoc networks. They defined the link stability as follows:

1. a link between two nodes A and B with transmission range r is established at time instant t_b when the distance between both nodes, defined as $d(A,B)$, is such that $d(A,B) < r$;

2. a link between two nodes A and B with transmission range r is broken at instant time t_e when the distance between both nodes is $d(A,B) > r$;

3. the link age a between two nodes A and B is the duration defined as $a(A,B) = t_f - t_b$.

The link stability is computed using a statistical-based approach, in order to discriminate among many links which are more stable through the estimation of the residual lifetime of each link. The expected residual lifetime $R_{A,B}(a_{A,B})$ of a link (A,B) is defined and computed from gathered statistical data as follows:

$$R_{A,B}(a_{A,B}) = \frac{\sum_{a=a_{A,B}}^{a_{max}} a \cdot d[a]}{\sum_{a=a_{A,B}}^{a_{max}} d[a]} - a_{A,B} \tag{5.2}$$

The value a_{max} is the maximum age reached by a link from the subject node point of view. Vector $d[\,]$ stores the observed links age, and element $d[a]$ represents the number of links with age equal to a.

Based on the expected residual lifetime R, a stability coefficient $s_{A,B}$ was defined to be exploited in the metric computation. It is defined as follows:

$$s_{A,B} = \frac{d_{A,B}^{avg}}{R_{A,B}(a_{A,B}) \cdot k} \quad \forall (A,B) \in E \tag{5.3}$$

The set E includes all the links between nodes in the network, $d_{A,B}^{avg}$ is the average distance between nodes A and B computed over the time they remain within transmission range. The scaling factor k is defined to make coefficient s comparable with other metrics.

A modification was needed to "tailor" the stability coefficient to this proposal. In the original work, it was used as a metric, so its value can be any positive real number, with lower values stating better link stability. To compare the link stability to the trust value, as we did in Eq. (5.1), we defined LSI in Eq. (5.4), in order to have values between 0 and 1,

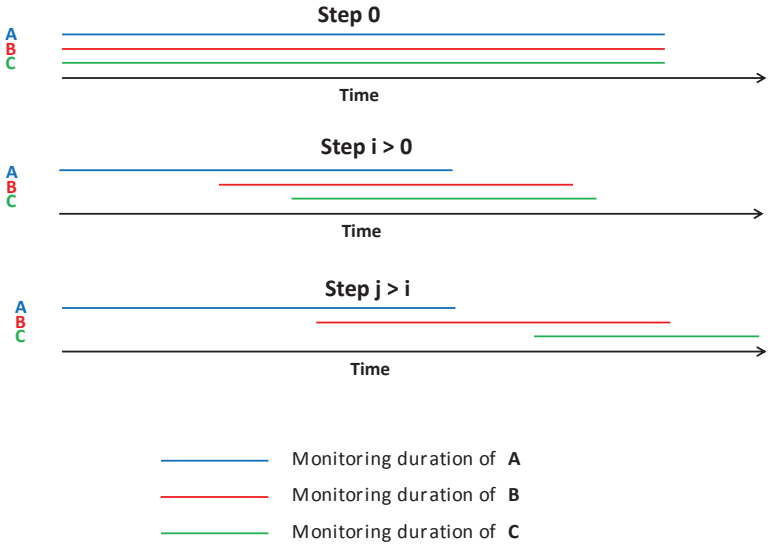


Fig. 5.2 Time division distributed monitoring

which represent the worst and best values, respectively.

$$\mathrm{LSI}(A,B) = \frac{r - d_{A,B}^{avg}}{r} \cdot \frac{2 \cdot R_{A,B}(a_{A,B})}{a_{max}+1} \quad \forall (A,B) \in E \tag{5.4}$$

The maximum value of $d_{A,B}^{avg}$ is r, so the first part of LSI formula has the value of 0 when $d_{A,B}^{avg}$ is equals to r, meaning that the nodes are far away, while the value tends to 1 when the nodes are close. The second part of LSI formula concerns the link duration. Having longer expected residual lifetime means more reliable links between nodes, so it has values near 1 when $R_{A,B}(a_{A,B})$ has higher values, otherwise its value is near 0.

5.2 Distributed monitoring

In this proposal, network participants start by jointly monitoring the network from the beginning. Afterward, the network monitoring task will be distributed among peers trusting each other, with the main purpose of saving energy when CNs perform monitoring, as shown in Fig. 5.2.

In order to make the monitoring process fully distributed among honest peers, some information needs to be exchanged among them. The computation of monitoring period is based on two values:

- Monitoring Duration (MD);
- Monitoring Starting Time (MST).

After a certain period (certain number of interactions) the trust table of every node will be updated. Thus, every node will start trusting a certain number of neighbors, which could be CNs. At this stage the nodes can start distributing the monitoring process among them by sharing their MDs along with the MST by sending these values periodically, through a new type of packet or attaching them to a packet already provided by the exploited routing protocol (e.g. Hello messages of AODV). Every node computes its MD using the amount of CNs and the duration of the links between them, by lowering the MD time when the node has more CNs. Equation (5.5) illustrates MD computation details.

$$\mathrm{MD}(A) = \begin{cases} 0, & \text{if} \quad \#C = 0 \\ \frac{\sum_{n \in N} R_{A,n}(a_{A,n}) \cdot \#N \cdot (\#C+1)}{(\#C)^2}, & \text{otherwise} \end{cases} \tag{5.5}$$

The coefficients of MD formula are explained in the following:

- N is the set of all the neighbors of node A;
- $\#N$ is the cardinality of set N;
- $\#C$ is the cardinality of CT;

The value of MD coefficient is determined by the Companion to Neighbor Ratio (CNR):

$$\mathrm{CNR} = \frac{\#C}{\#N} \tag{5.6}$$

The higher is the value of CNR, the lower is the monitoring duration. If a node has not any CN, two scenarios can be distinguished:

- the node is isolated, so it could avoid to keep the wireless interface in promiscuous mode, hence saving energy;
- nodes in the neighborhood are not trusted or still not evaluated, so the node should keep monitoring until the next MST computation.

Table 5.1 Simulation Parameters

Parameter	Value
Area	1000×1000 m^2
Duration	10 minutes
Source nodes	5
Transmission rate	4 packets/s
Data packet size	256 bytes
Transmission range	200 m
CS_{th}	0.3
α	0.6

When a node receives different MDs and MST values from its neighbors, it computes its next MST in the way described by the following equation:

$$\mathrm{MST}(A) = \max_{n \in N}[\mathrm{MST}(n) + \mathrm{MD}(n)] \tag{5.7}$$

Once a node detects a malicious agent, which means that the agent trust value goes below a detection threshold T_{th}, it broadcasts a one-hop negative recommendation that includes the detected node identity. The node A that receives this recommendation will autonomously put the malicious agent in blacklist for a period $t_{\mathrm{blacklist}}$ computed as follows:

$$t_{\mathrm{blacklist}} = \frac{\log \varepsilon}{\log \rho} \cdot \mathrm{CS}(A,B) \tag{5.8}$$

Equation (5.8) is inspired to Eq. (4.4). The blacklist period is longer for higher values of CS, which means that the node A trusts the recommendation of node B depending on its trust value T and its link stability defined by LSI.

5.3 Performance evaluation

To evaluate the performances of this approach concerning distributed monitoring, it is implemented on AODV routing protocol. Many simulations were run using the Network Simulator 3 (NS-3) simulator. The parameters which are common to all simulations are illustrated in Table 5.1. The total number of nodes varies from 50 to 300, with pedestrian speeds from 0 to 3 m/s. The α value is taken from [80]. The amount of malicious nodes varies from 10% to 20% of the nodes. The exploited attacker model is described in Section 4.3, with dropping probability d of 50% or 100%, defined for each graph.

Table 5.2 Energy Model Parameters

Parameter	Value
$m_{\mathrm{Rx}_{\mathrm{prom}}}$	0.388 mJ/byte
$b_{\mathrm{Rx}_{\mathrm{prom}}}$	136 mJ
$b_{\mathrm{discard}_{\mathrm{ctl}}}$	0 mJ

The main focus of this proposal is the reduction of the energy consumption by distributing monitoring operations, which use the promiscuous mode of the wireless interface, among trusted neighbors. Therefore, we exploited the linear energy model illustrated in Section 4.2. Specifying the model to promiscuous mode cost, the following equation, based on Eq. (4.20), represents the cost of each packet for nodes which are monitoring the transmission:

$$\mathrm{Cost}_{\mathrm{no\_dst}} = \sum_{n\in S} b_{\mathrm{discard}_{\mathrm{ctl}}} + \sum_{n\in D} b_{\mathrm{discard}_{\mathrm{ctl}}} + \sum_{n\in S} (m_{\mathrm{Rx}_{\mathrm{prom}}} \times \mathrm{size} + b_{\mathrm{Rx}_{\mathrm{prom}}}) + \sum_{n\in D} b_{\mathrm{discard}_{\mathrm{ctl}}} \tag{5.9}$$

Since the analysis only takes into account the energy consumption due to monitoring activities, the parameter values in Eq. (5.9) are defined in Table 5.2 as incremental differences with respect to the idle mode energy consumption of the wireless interface.

The accuracy of the IDS is represented by the detection ratio, defined as follows:

$$\text{Detection ratio} = \frac{\#true\_positives}{\#detections} \tag{5.10}$$

Fig. 5.3 shows the detection performances of the proposal in different scenarios, allowing to analyze the ratio of true positives with respect to the total amount of detections over time. After about 300 seconds, the true positives percentage seems to be stable, with a better accuracy in networks with less nodes. In the best scenario, with 50 nodes, 10 malicious nodes and $d = 50\%$, the accuracy grows above 80%, while for higher drop probabilities (see Fig. 5.4), performance deteriorates a little. For each scenario, the true positives percentage varies between 0% and 10% less than the results with $d = 50\%$.

The energy consumption of the proposal is analyzed in Fig. 5.5. The analysis concerns the scenario with 10% of malicious nodes and $d = 50\%$, because other parameter combinations provide nearly the same results when fixing the amount of nodes. The trend of each curve is almost linear over time. The lowest energy consumption is obtained with 50 nodes, reaching an promiscuous mode energy consumption of about 1200 J. When increasing the amount of nodes, thereby having more source nodes and more packets in the network, the energy

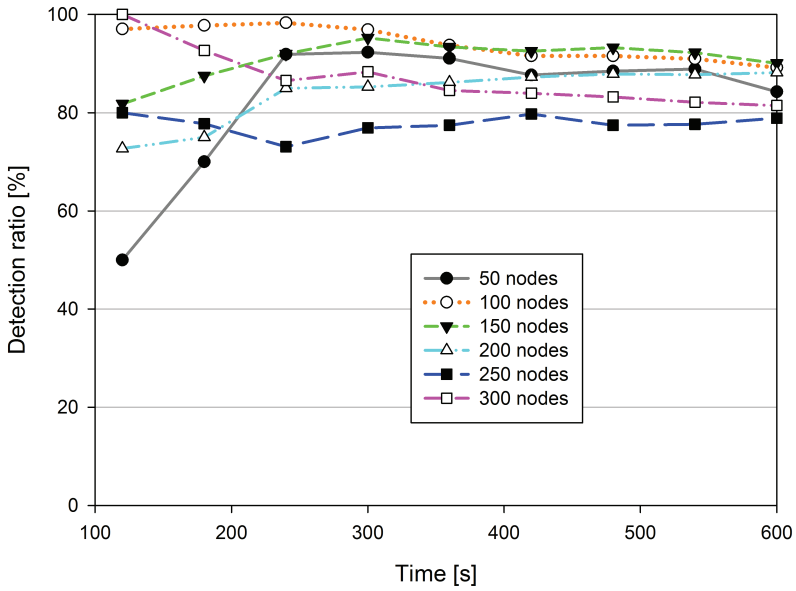


Fig. 5.3 Detection performances with 20% malicious nodes ($d = 50\%$)

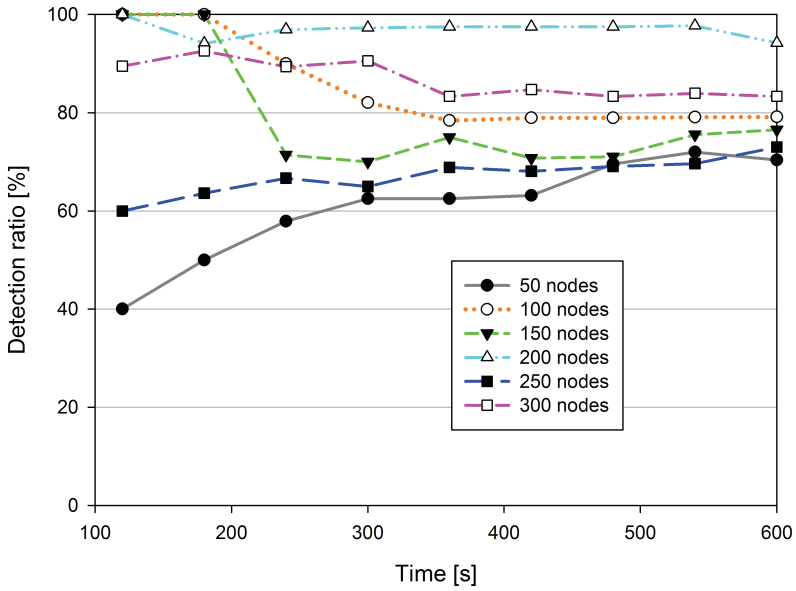


Fig. 5.4 Detection performances with 20% malicious nodes ($d = 100\%$)

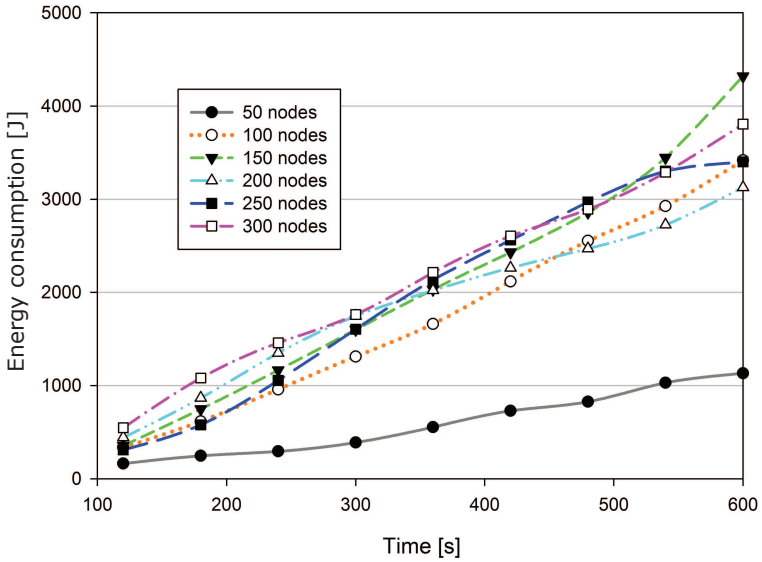


Fig. 5.5 Energy efficiency with 10% malicious nodes and $d = 50\%$

consumption increases as well. Having more nodes in a scenario with the same size causes that more packets are sensed through the promiscuous mode of the wireless interfaces of the nodes. However, thanks to the distributed monitoring strategy exploited, the consumed energy remains nearly the same for high node densities, thus showing a huge optimization compared to the normal scenario where a higher number of nodes leads to an increase in the energy consumption

The PDR, computed as in Eq. (4.23), shows the capability of the protocol to deliver packets in the presence of malicious nodes. The results under this perspective are shown in Fig. 4.7. PDR improves when the network has a higher number of nodes. For most scenarios studied, the PDR is stable after 300 seconds, while for 50 and 100 nodes (where PDR has the lowest values) we find a slightly increasing trend after 300 seconds as well. With more than 100 nodes, the PDR is about the 90%, which is a good result considering that 20% of the nodes drop half of the packets received. This means that most attacks are detected, and so misbehaving nodes are excluded from the valid routes connecting the nodes in the network.

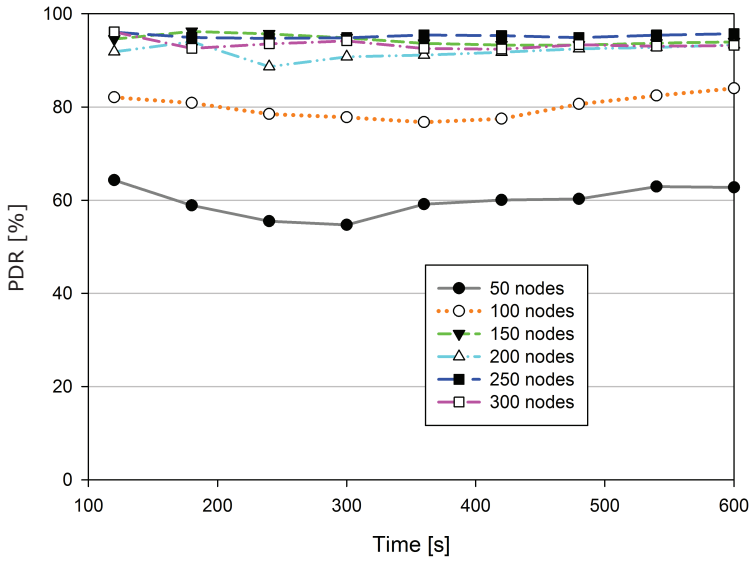


Fig. 5.6 PDR with 20% malicious nodes ($d = 50\%$)

Chapter 6

Trust Model Enhancement through Probabilistic Monitoring for Improving Energy Consumption of Intrusion Detection System

In trust-based approaches for intrusion detection, the trustworthiness of a node can be evaluated by monitoring its actions during the communication. A node can evaluate which agents are more trustworthy, therefore they can be chosen to be part of the route toward the destination. Reducing the amount of monitoring operations leads to an improvement in terms of energy saving for the nodes composing the network. Monitoring cannot be avoided, because the behavior of nodes could change over time due to malicious agents that could compromise fair nodes and selfish behaviors due to energy saving reasons. A probabilistic approach to reduce monitoring operations is proposed, showing that linking the trust value of an agent to the probability to monitor the interaction with it can lower the energy consumption of the IDS in wireless ad-hoc networks.

Many models were defined and used for this proposal. Trust model shows the way in which the trust value of agents is computed. The exploited trust model for this proposal is described in Section 4.1. The energy model gives an idea about the consumption due to transmission, monitoring and cryptography operations. In resource constrained environments such as MANETs, energy is one of the main issues to be faced. Each operation has an associated cost, depending on the size of the packet and on the nodes involved. For the purposes of this proposal, the energy model adopted is illustrated in Section 4.2. The attacker model, described in Section 4.3, is defined to introduce a threat in the network, discarding

Table 6.1 Models Symbols

Symbol	Meaning
T	Trust value
ρ	Remembering factor
t	Time
ε	Lowest weight for an observation to be accounted
R	Transmission rate
PMF(T)	Probabilistic Monitoring Function for T value
$T_{\min}$, $T_{\max}$	Minimum and maximum values of trust
T_{th}	Threshold trust value
Beta$(x;p,q,a,b)$	Beta distribution for x value
a, b	Beta distribution lower and upper bounds
p, q	Beta distribution shape parameters
B(p,q)	Beta function with parameters p and q
β-PMF(T)	PMF based on Beta distribution
d	Packet drop probability
m_{action}	Incremental cost of *action* in energy model
b_{action}	Fixed cost of *action* in energy model

a part of received packets following a probability. The nodes executing an attack should be detected by the fair nodes using the trust and monitoring models. Monitoring model, which is the core of this proposal, defines how monitoring operations will be reduced using a Probabilistic Monitoring Function (PMF) [81, 82], so a fair node is supposed to be less monitored than a malicious one, with a consequent energy saving.

6.1 Monitoring model

The meaning of monitoring is to observe and check the progress or quality of some network parameters over a period of time. In ad-hoc networks, the monitoring aim is securing the routing operations, so nodes behaving maliciously or selfishly will not disrupt the paths between the nodes. Using the promiscuous mode to monitor packets that need to be forwarded is an expensive operation, so an energy-efficient approach has to reduce the monitoring activity when it is not needed [28], namely when the interactions occur among fair nodes. A simple idea to address this issue is the linkage of the monitoring probability to the trustworthiness of the agent. The network status and the transmission characteristics must be taken into account too. In Table 6.1 a summary of the symbols used in the proposed model and in the following of this Chapter is given.

PMF is defined to link the trust value to the probability of monitoring the action requested. This function has to respect the following properties:

$$0 \leq \text{PMF}(T) \leq 1, \quad \text{for} \quad T_{\min} \leq T \leq T_{\max} \tag{6.1}$$

$$\text{PMF}(T) = 1, \quad \text{for} \quad T \leq T_{\text{th}} \tag{6.2}$$

$$\text{PMF}(T_1) \geq \text{PMF}(T_2), \quad \text{for} \quad T_1 < T_2 \tag{6.3}$$

The term T_{th} is the trust value that an agent has to pass to be trusted, T and T_i are trust values included between $T_{\min}$ and $T_{\max}$, which are are respectively the minimum and maximum trust value allowed.

PMF properties can be described as follows:

1. it represents a probability, so its value must be included between 0 and 1 for each allowed trust value, as defined in Eq. (6.1);
2. when the agent is unknown or distrusted, the monitoring probability must be 1, represented by Eq. (6.2);
3. the function must be monotonically decreasing, as in Eq. (6.3).

The PMF shape needs to be changed dynamically according to the status of the network and the characteristics of the transmission, so it should be parametric. A function that could fulfill the properties needed for the PMF with just some adjustments is the well-known Beta distribution [83].

$$\text{Beta}(x; p, q, a, b) = \frac{(x-a)^{p-1}(b-x)^{q-1}}{\text{B}(p,q)(b-a)^{p+q-1}}, \quad \text{for} \quad a \leq x \leq b;\ p, q > 0 \tag{6.4}$$

This distribution is defined in the range of values [0, 1], having four parameters, which are:

- p and q determining the shape;
- a and b respectively as lower and upper bounds.

$\text{B}(p,q)$ is the Beta function, defined as follows:

$$\text{B}(p,q) = \int_0^1 t^{p-1}(1-t)^{q-1}\text{d}t \tag{6.5}$$

Distribution functions are monotonically increasing. Exploiting this property, the β-PMF can be defined as follows:

$$\beta\text{-PMF}(T) = \begin{cases} 1, & \text{for } T < T_{\text{th}} \\ 1 - \text{Beta}(T; p, q, T_{\text{th}}, T_{\text{max}}), & \text{for } T \geq T_{\text{th}} \end{cases} \tag{6.6}$$

Equation (6.6) fulfills all the properties defined for the PMF. The monitoring probability depends on the trust value of the agent, through the p and q parameters the shape of the function can be dynamically changed. Beta distribution was chosen as basis for a concrete implementation of the PMF because its properties were deeply analyzed in literature. Furthermore, the availability of four parameters defining its shape could be exploited adapting the function to changing security requirements in a very dynamic environment as in MANETs.

Algorithm 2 Probabilistic monitoring algorithm

```
for packet = packet sent to neighbor n do
  m = PMF (T_n)
  rnd = random generated number
  if x ≤ m then
    MONITOR packet
  else
    IGNORE packet
  end if
end for
```

6.1.1 Integration with trust model

The trust modeling framework defines which are the values of T_{th}, T_{min} and T_{max}. For the adopted model, shown in Section 4.1.1, the values of T_{min} and T_{max} are respectively -1 and 1. The threshold represents the highest uncertainty value about the action execution, namely $T_{\text{th}} = 0$. The parameters p and q depend on the status of the network and the characteristics of the transmission (e.g. transmission rate, bit error rate). Evaluating the Fig. 6.1, the constraint $p \geq q$ is defined to avoid a sudden decrease of the monitoring probability for trust values near T_{th}.

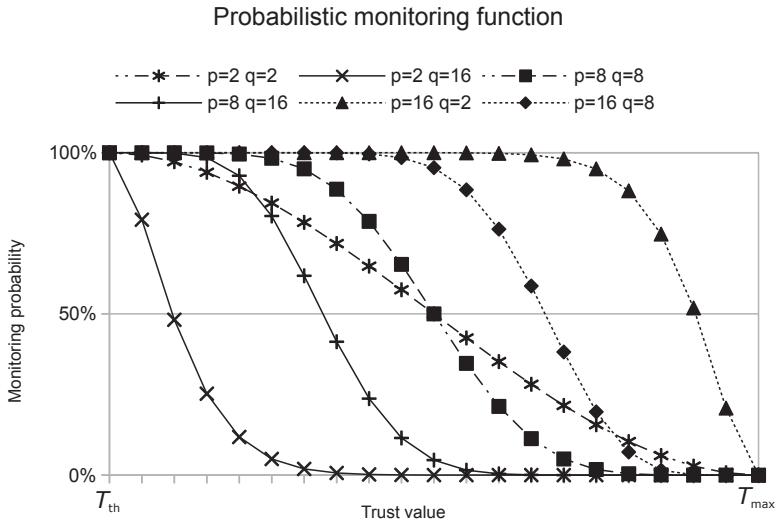


Fig. 6.1 Shape of β-PMF for various p and q values

6.2 Analytical evaluation

As first step to evaluate the proposed monitoring model, a network composed of a source node, a destination node and n intermediate nodes having a path toward the destination is introduced. A graphical representation of this network topology is shown in Fig. 6.2. The source sends the data packets to a neighbor at a given rate until its trust value is above T_{th}. If the chosen node becomes distrusted, the source chooses another neighbor as intermediate node. In Table 6.2 the various used parameters are shown. The values m and b are applied to the linear energy model presented in Section 4.2. Many runs using the same

Table 6.2 Parameters

Parameter	**Value**
Packets transmitted	1200 packets
Packet size	512 bytes
R	{0.5, 1, 2, 4} packets/s
d	{0%, 25%, 50%, 75%, 100%}
p	16
q	{2, 4, 8, 16}
ρ	{0.666, 0.8, 0.9}
$m_{Rx_{prom}}$	0.388 mJ/byte
$b_{Rx_{prom}}$	136 mJ
Runs	100

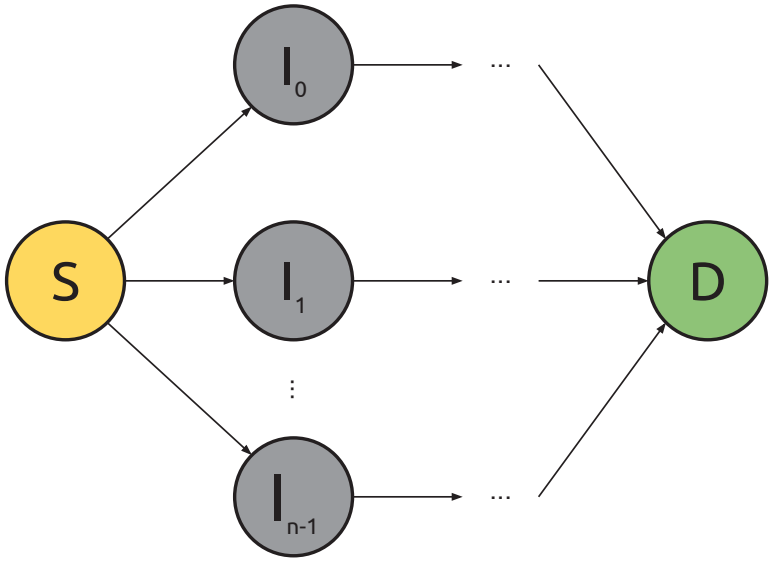


Fig. 6.2 Network scenario for analytical evaluation

parameters were executed to compute the 95% confidence interval on the figures where they are shown. The channel error model follows a standard uniform distribution, the probability with which the packet transmitted to the intermediate node is lost/corrupted is of 2%. Let ε be the threshold value below which the observation does not contribute to the trust value computation, as defined in Eq. (4.3). The values of ρ_t were chosen in order to have the weight less than $\varepsilon \leq 0.1\%$ for times further than t with $t \approx \{15, 30, 60\}$ s, resulting respectively in $\rho = \{0.666, 0.8, 0.9\}$. The parameter d represents the probability that a malicious node will drop the received packet, as defined in Section 6.1. The evaluated results concern the accuracy of the intrusion detection and the energy saved by the probabilistic monitoring. The detection accuracy is evaluated through the time needed to detect a malicious or selfish agent, the amount of false positives detected in the system and the detection effectiveness. Probabilistic monitoring allows reducing the costs concerning the energy consumption due to the promiscuous mode used to monitor the correct forwarding of the packets toward the destination, so these costs are taken into account. Analyzing the results data, a similarity can be noted when the ratio between p and q has the same value, so p value was fixed to 16 in order to have many different integer values of q that fulfill the constraint $p \geq q$ defined in the model. In the following figures, standard monitoring means the application of the TMS without PMF, while β data series indicate the results obtained using the proposed monitoring function.

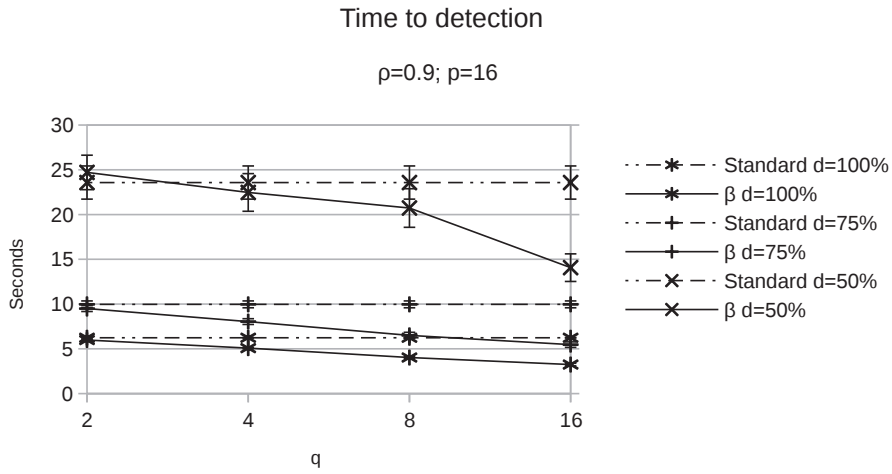


Fig. 6.3 Time needed to detect malicious nodes with $\rho_{60} = 0.9$

6.2.1 Time to detection

The main aim of an IDS is the protection of the network against malicious behaviors by detecting the threats in a timely manner. The evaluation of the time needed for detection with respect to the standard monitoring is done using different values for ρ and q, examining the variations they introduce. The transmission rate R was fixed to 4 packets per second, the malicious nodes behavior was changed varying d value from 100% to 50%. When the drop percentage is the 25%, the monitoring works only under some conditions, therefore the analysis of this scenario is done separately. The malicious node behavior model is developed to keep a fair behavior without dropping packets for the first 60 seconds of simulation, beginning to drop packets since this time. Fig. 6.3 shows the time needed to detect a node that begins to drop packets. With the value of $\rho_{60} = 0.9$, the trust value computation takes into account the last 60 seconds of interactions between the subject and the agent. The standard curve keeps a constant value because it is independent of the β-PMF parameters. For higher values of q, probabilistic monitoring achieves better results than the standard monitoring. It could look like a contradiction, but the reason can be simply explained. The trust value is time-dependent, so reducing the monitoring activity after a certain threshold value defined by the parameters of the PMF keeps the trust values generally lower than the standard scenario. Starting from a lower value allows the IDS to react promptly to the threat, detecting it faster. When $q = 16$, so $p = q$, PMF achieves a time saving between the 40% and the 48% of the time needed with respect to the standard monitoring. For lower values of q the time of detection increases. The same trend can be observed with lower drop percentages,

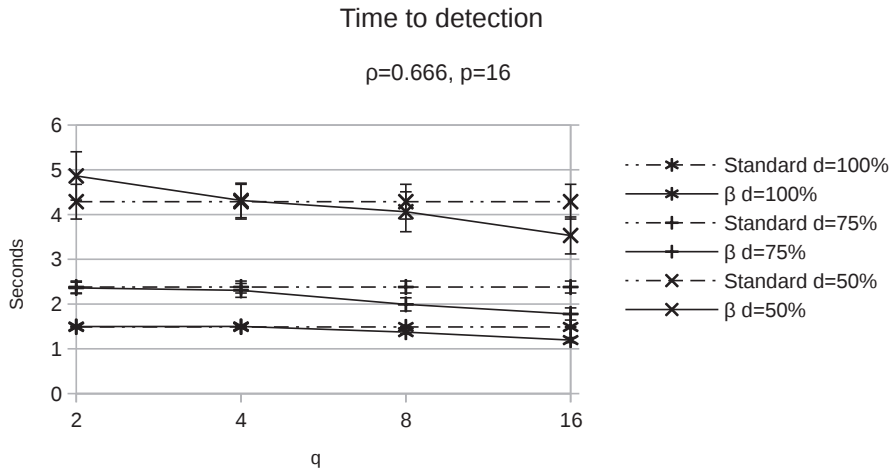


Fig. 6.4 Time needed to malicious node detection with $\rho_{15} = 0.666$

because a node that drops less packets is harder to detect. Probabilistic monitoring achieves a better result than the standard monitoring in all cases except with $d = 50\%$ and $q = 2$, where the standard monitoring detects the threat with about 5% less time. Using this combination of parameters leads to generally higher trust values, similar to the standard monitoring, but the percentage of unmonitored packets nullifies the responsiveness of the PMF achieved in other studied cases. When only 15 seconds are taken into account in the trust value computation (using $\rho_{15} = 0.666$), the reaction to the threats is faster, and the time needed to detection is lower than the previous case. The results with this configuration are shown in Fig. 6.4.

The saving in terms of time is limited, between the 17% and the 25% with respect to the standard monitoring. The trend is similar to the previous graph, the motivations are almost the same as before. In the terms of time needed to detection, the energy-efficient monitoring obtains worse results than the standard monitoring only when the drop percentage is 50% and $q \leq 4$, while with lower drop percentages, the curves reach at most the value obtained with the standard monitoring when q has the lowest value.

6.2.2 Detection accuracy

Standard and probabilistic monitoring are both able to detect the malicious behavior of a node that drops the 50% or more of the total number of received packets. For lower percentages, the detection is harder, therefore PMF parameters need to be tuned consequently. The results of the various simulations performed to understand the effect of the parameters values on the detection are shown in Fig. 6.5.

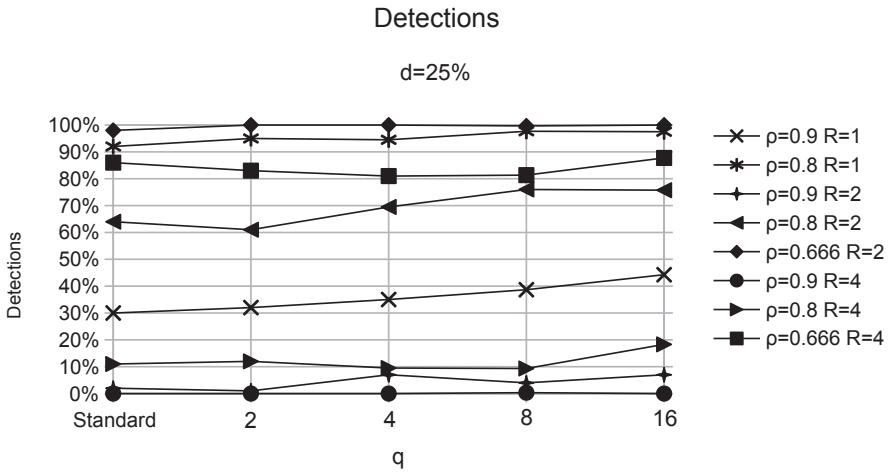


Fig. 6.5 Malicious node detection with $d = 25\%$

The transmission rate is represented by R, in packets per second. The results show that the parameters p and q have a small effect on the accuracy, but for higher values of q it seems to improve a little. With the q highest value, the accuracy of the energy-efficient monitoring is always better than the accuracy of the standard monitoring. The characteristics that have a huge effect on the detection accuracy are the remembering factor and the transmission rate. Lower values of ρ_t and R allow the detection of small drop percentages. The relation between the rate R and the time t, which depends on ρ_t as in Eq. (4.4), can be defined as follows:

$$R \cdot t \leq 60 \tag{6.7}$$

The above equation should be respected to have a high accuracy in positive detections (with a percentage $\geq 90\%$) when the packet drop rate of the misbehaving nodes is the 25%. Combinations of ρ_t and R that satisfy Eq. (6.7) are capable to detect this kind of malicious behavior.

The detection accuracy depends also on the ability to detect only misbehaving nodes, excluding the nodes that do not execute the required action due to nodes mobility and the use of a wireless transmission channel, which could lead to errors in packet transmission. Fig. 6.6 shows that the erroneous detections concerning fair nodes detected as malicious do not depend on the probabilistic monitoring, keeping almost the same trend for the standard monitoring and the various values of q when using PMF. The analysis of the false positives allows to define another relationship between the rate R and the time t that needs to be

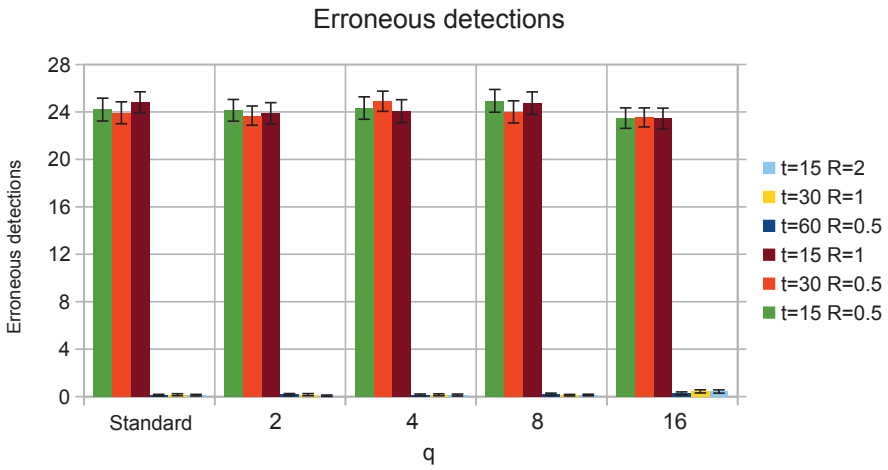


Fig. 6.6 False positive detections

satisfied in order to avoid most of the erroneous detections, which is the expected behavior.

$$R \cdot t \geq 30 \tag{6.8}$$

6.2.3 Energy consumption analysis

In Section 4.5.4, the outcome of the energy consumption analysis shows that it is included between the 28% and the 41% of the overall energy consumption in various network conditions, while these percentages are respectively the 34% and the 39% when all the nodes in the network behave fairly. The aim of the probabilistic monitoring is the improvement of the effectiveness of the system and the life of the nodes participating in the network by reducing the monitoring activity.

The energy consumption reduced by using the probabilistic monitoring, as shown in Fig. 6.7. The saved energy is the direct consequence of the reduced number of monitored packets. Increasing the ρ_t value reduces the consumption, because the average value of trust for fair nodes is higher when more observations contributes to trust computation, allowing less monitoring operations. The consumption is higher for lower q values, because more packets are monitored for lower trust values. In the best case (with $q = 16$ and $\rho_t = 0.9$) the saved energy is the 70% of the monitoring energy consumed while using the standard monitoring. The consumption reaches almost the same energy consumed by the standard monitoring when $q = 2$, because PMF values are near the 100% of the monitored packets for trust values lower than 0.7.

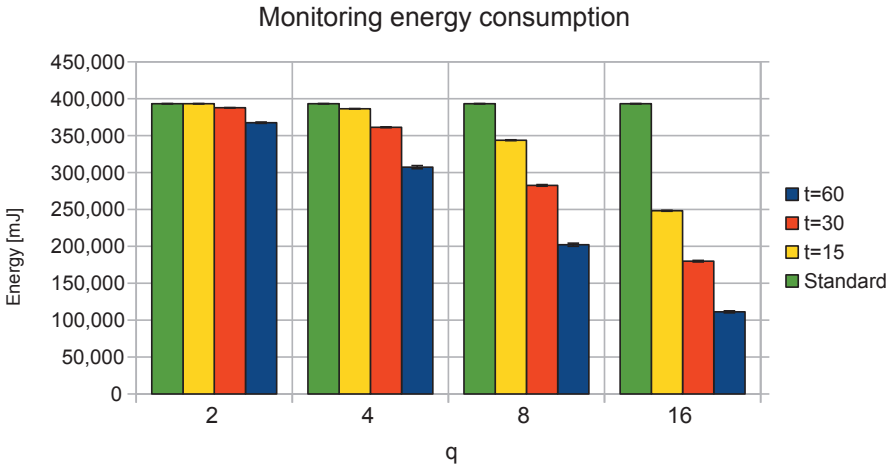


Fig. 6.7 Energy consumption of monitoring operations

6.3 STAMP implementation in NS-3

In order to better evaluate the proposed approach, it was implemented in NS-3.26 [84] by developing a module that allows its use in many routing protocols for wireless ad-hoc networks with just few modifications. This choice was made to analyze the proposal performance by executing it in a realistic and well-known simulation environment. The module design and its interactions with NS-3 modules are illustrated in Fig. 6.8. The main components of the TMS module are:

- Trust Manager (TM), which is responsible of managing all the events needed to compute trust values;
- Trust Table (TT), composed of a collection of Trust Table Entries (TTEs) to store the results of the interactions with other nodes. TT notifies the node if an agent becomes distrusted, and it is asked by the node if it needs to know if a node is trustworthy;
- PMF, an implementation of the β-PMF illustrated in Section 6.1.

All the transmission and reception events are notified to the TM, exploiting the callback API for the promiscuous reception and the tracing system for transmission events. The way in which the routing protocol is aware of the trust values of its neighbors depends on the specific protocol used. It is obtained through calling directly the TT when the protocol needs to verify if a neighbor node is trustworthy, while a callback is used when a node becomes untrustworthy. Testing the proposal required the implementation over an existing routing

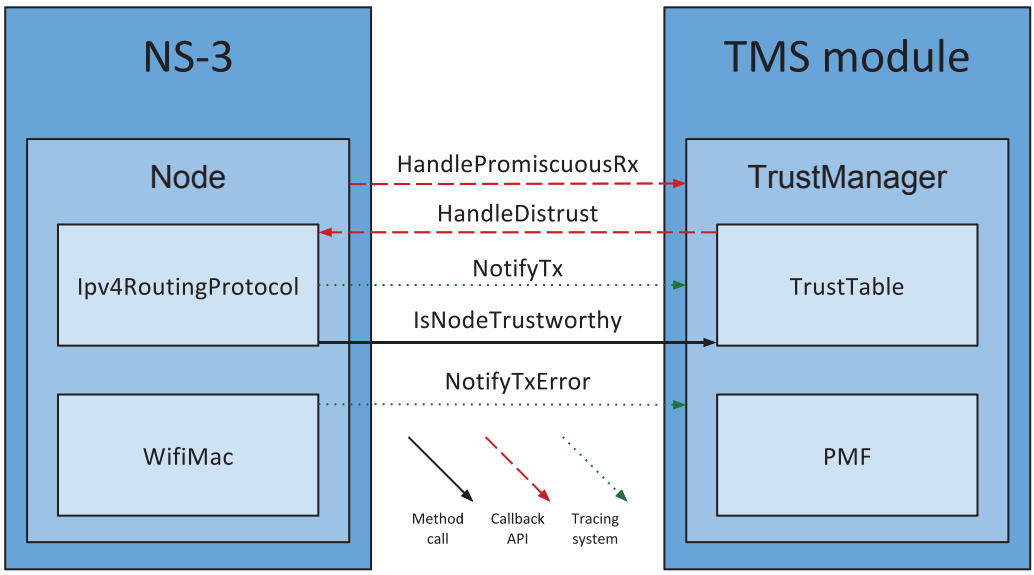


Fig. 6.8 TMS module in NS-3.26

protocol in NS-3, therefore the TMS module was integrated with SAODV protocol, in its turn developed from AODV protocol available in the simulation framework. The integration of TMS with SAODV protocol takes the name of Secure and Trusted AODV Monitored Probabilistically (STAMP). TMS module implementation is protocol-independent, so it can be integrated with any of the available protocols in NS-3 with a little effort. The code of the module is available in Appendix A of this thesis.

6.3.1 Trust Manager

TM is the core of the module. It initializes TT and PMF, calling the first class when a new observation is done, and both of them when it takes the decision of monitoring a packet or not. TM registers as trace sink for sent packets and manages all promiscuous receptions, so it checks if neighbor node forwards a packet correctly by monitoring the transmission. Sent packets are kept into a buffer until forwarding is detected or a timeout expires. These two events result in a new observation (positive or negative) that is registered in TT.

6.3.2 Trust Table

The purpose of the TT is keeping track of all the observations done by a node, so it can be enquired when the protocol needs to know if an agent is trustworthy or not. When a node becomes distrusted, a callback is triggered to notify the node about the event, so the protocol

can take the appropriate measures against the agent. TT acts as trace source for distrusted and trust change events, so an external module can connect to these traces. Each agent is registered in a TTE the first time an interaction occurs. Each TTE is identified by the agent and the action. It contains the values needed for computation of P as defined in (4.2), which are numerator and denominator values, the time in which last interaction with the agent occurred and its actual trust value.

6.3.3 Probabilistic Monitoring Function

PMF is used each time a transmission is made and TM needs to decide if it will be monitored or not. It needs the two parameters p and q to define the Beta distribution to use in its computation. It requires the trust value in input, hence it returns the value of the monitoring probability for that trust value, using Eq. (6.6). The function is called each time a packet is sent, its result is compared with a random generated number computed between 0 and 1, deciding if the packet will be monitored or not, as described in Algorithm 2. PMF implementation exploits the Beta distribution by using Boost libraries [85].

6.3.4 Routing Protocol

TMS and attacker model implementations require some modifications to original AODV module provided by NS-3. Concerning the TMS, modifications were made on initialization of the protocol and management of route requests. During its initialization, the protocol registers the method defined to manage links break with the neighbors as *HandleDistrust* callback for TMS, as shown in Fig. 6.8. Therefore, the distrust event of a neighbor is managed as a common link break between two nodes, setting invalid the routes with the distrusted agent as next hop and sending a route error message to all precursor nodes in those routes. The only change done to the route request reception management concerns checking if the request is received from a trustworthy node. If the sender is evaluated as malicious, the route request message is ignored. Malicious behavior is implemented in *RouteInput* method of the exploited protocol. In this method, all non-protocol packets are managed. Each malicious node computes a random number between 0 and 1, comparing it with its d value and choosing to forward the packet or not based on this test.

Table 6.3 Trust and Monitoring Model Parameters

Parameter	Value
p	{8, 16}
q	{2, 4, 8, 16}
t	{15, 30, 60} s
ε	10^{-3}

6.3.5 Cryptography

The implementation on SAODV protocol required the use of cryptography on AODV protocol, because NS-3 simulator does not provide it. The routing table was enhanced to store the fields needed by DSE. SAODV computes and adds the signature extension to all protocol packets: RREQs and RREPs allow both SSE and DSE, RERRs requires only the SSE. When a protocol packet is received, the signature is verified. If the signature is missing or invalid, the packet is discarded. For each type of packet, the required signature extensions were implemented. More details about SAODV are available in Section 3.3.4. All cryptographic operations were implemented exploiting Crypto++ library [86].

6.4 Performance evaluation

The implementation of TMS in NS-3 enabled the evaluation of the proposal exploiting the already available protocols and modules. Nodes move in an area without obstacles following RWP mobility model [77]. At the beginning, each node is placed randomly in the area. Each source node sends packets to a destination node at a certain rate, starting after 60 seconds of simulation. Malicious nodes follow the attacker model illustrated in Section 4.3. The comparison of STAMP is done against AODV and SAODV protocols. Concerning the monitoring model, in many figures the ratio between q and p is taken into account, because their ratio influences the performance more than their absolute values. The cryptographic algorithms used in SAODV and STAMP are RSA and SHA1 respectively for signatures and hashes. Trust and monitoring model parameters are shown in Table 6.3, values used in attacker and mobility models are illustrated in Table 6.4, while all the values assigned to other parameters used in the simulation campaigns are shown in Table 6.5. The values assigned to the energy model parameters, shown in Table 6.6, represent the variation among the action executed by the wireless interface and its idle state, so the incremental cost of discarding a packet is negative because there is an energy saving with respect to the idle state.

Table 6.4 Attacker and Mobility Models Parameters

Parameter	Value
Malicious nodes	{0, 5, 10, 15}
d	{0%, 25%, 50%, 75%, 100%}
Max speed	5 m/s
Pause time	0 s

Table 6.5 Simulation Parameters

Parameter	Value
Time	600 s
Area	1000 m $\times$ 1000 m
Nodes	50
Source nodes	{5, 10}
Rate	{1, 2, 4, 8} packets/s
Packet size	256 bytes
Runs	30

Table 6.6 Energy Model Parameters

Parameter	Value
m_{Tx}	1.89 mJ/byte
b_{Tx}	246 mJ
m_{Rx}	0.494 mJ/byte
b_{Rx}	56.1 mJ
m_{discard}	-0.49 mJ/byte
b_{discard}	97.2 mJ

Parameter	Value
$m_{\text{Rx}_{\text{prom}}}$	0.388 mJ/byte
$b_{\text{Rx}_{\text{prom}}}$	136 mJ
$b_{\text{Tx}_{\text{ctl}}}$	120 mJ
$b_{\text{Rx}_{\text{ctl}}}$	29 mJ
b_{sign}	546.5 mJ
b_{verify}	15.97 mJ
m_{hash}	0.75 μJ/byte

Performance evaluation is done over many different perspectives:

- PDR: it is analyzed to evaluate the capability of delivering packets when the network is under attack;
- average hop count: it should give details on the changes that occur in the routes between nodes when the TMS is used;
- control packets analysis: it provides an overview of the effect of the protocol on the total amount of packets circulating in the network;
- detection accuracy: it is evaluated through true positives and false positives, comparing different configurations of STAMP with and without PMF;
- energy consumption: it is evaluated to understand the effect of PMF on monitoring operations and how this effect leads to advantages in energy saving.

In the following figures, STAMP protocol is intended when only the parameters t, p or q are specified. When only t is defined, the protocol is working with a constant monitoring probability of 100%. The confidence intervals of 95% are computed where shown.

6.4.1 Traffic analysis

The analysis of the traffic is done through the study of the percentage of delivered packets over all the sent packets, investigating how the routes change when TMS is working, with and without PMF. The expected behavior is a higher PDR, at the cost of longer routes between source and sink nodes.

In Fig. 6.9 the PDR of the compared protocols is shown. In every condition STAMP outclasses AODV and SAODV protocols, even when the network is free of malicious nodes, with a marked advantage when the drop percentage increases. While PDR decreases without TMS, STAMP can deliver almost the same percentage of packets for values of d between the 25% and the 75%, after a little decrease in performance when malicious nodes join the network. STAMP can deliver more than the 10% more packets than other compared protocols when $d \geq 50\%$, 15% more when malicious nodes drop all received packets. Probabilistic monitoring achieves almost the same results of constant monitoring, so it does not introduce a loss in performance. The small difference between STAMP with different parameters is due to t value, better performance is achieved with smaller values.

Average hop count is shown in Fig. 6.10. When the network is under attack, route length is higher for STAMP. It increases when more nodes have a malicious behavior, because it

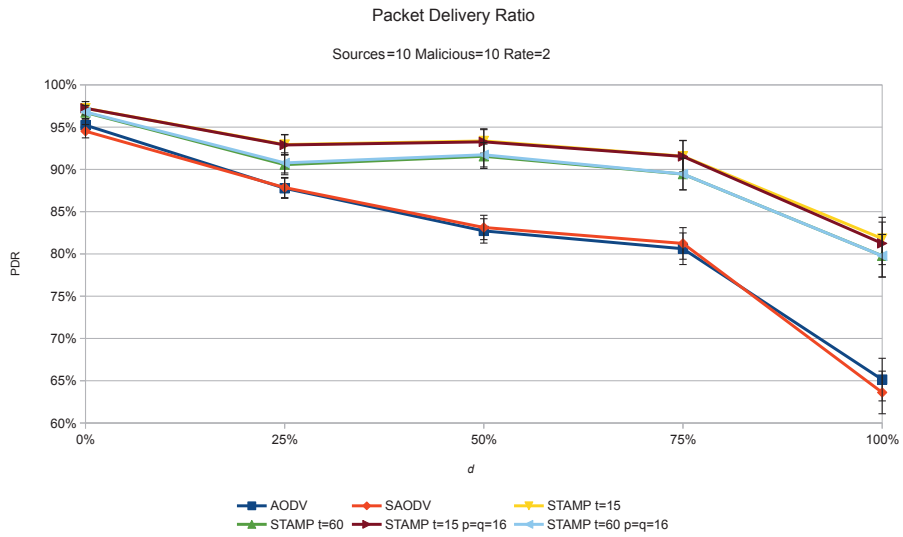


Fig. 6.9 Packet Delivery Ratio vs packet dropping probability

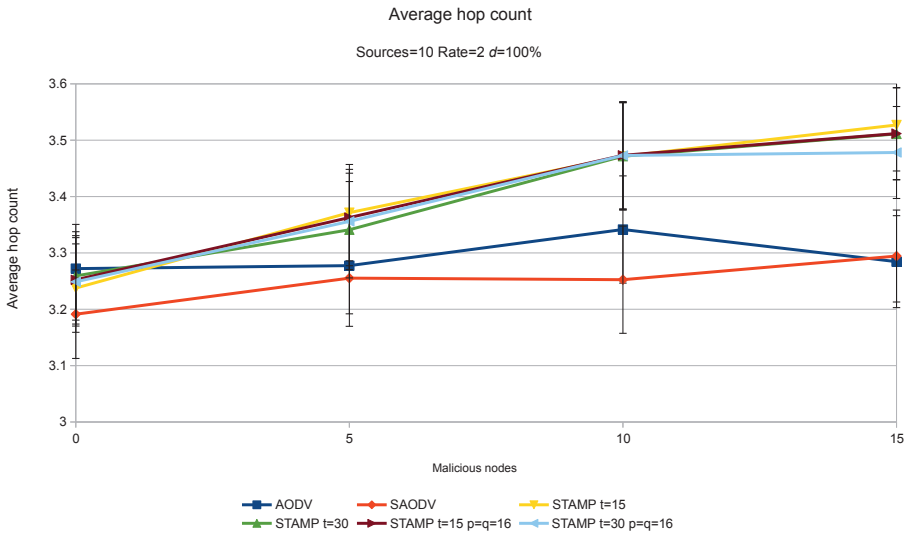


Fig. 6.10 Average hop count vs amount of malicious nodes

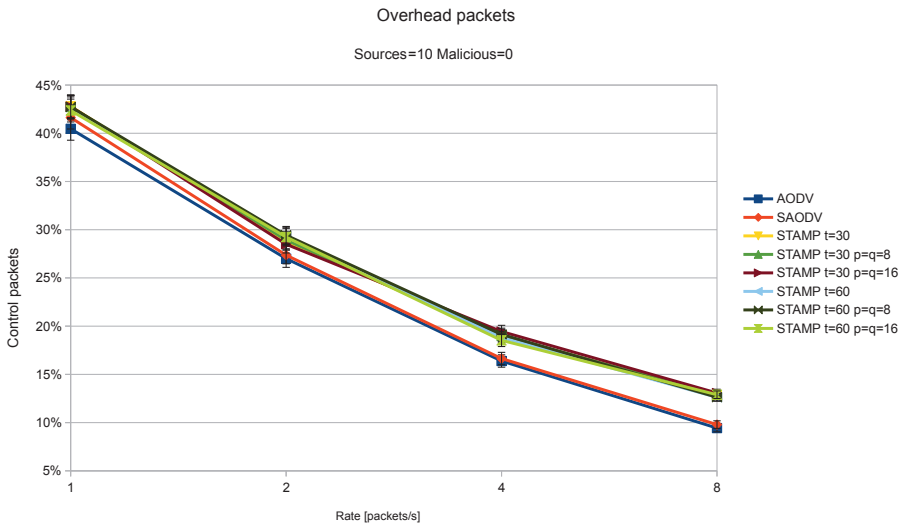


Fig. 6.11 Percentage of control packets when transmission rate increases

excludes detected nodes from the routes, so fair nodes can communicate avoiding them. In the defined scenario, with 50 nodes and an area of 1 Km^2, the increase in route length is low, because it is always included between 3 and 4 hops. For AODV and SAODV, average hop count is included between 3.2 and 3.3 hops for each number of malicious nodes in the network, because they cannot be detected so the routes are not affected by their presence.

STAMP protocol excludes detected nodes from the routes by issuing RERRs packets, so source node will search for another route. It occurs each time a new detection is done, because the protocol on which STAMP is based is reactive, without backup routes. The process could be done by the intermediate node that detects the threat, but the local repair mechanism is not mandatory and not widely supported. The analysis of overhead in the network is done due to these reasons.

The weight of control packets on the total amount of packets transmitted in the network is shown in Fig. 6.11. When no malicious nodes are in the network, the performance are similar between the protocols with and without TMS. Transmitting at higher rates allows to reduce the effect of the overhead, because for each established route, a higher amount of packets is transmitted. The percentage of control packets decreases of more than the 30% when the rate increases from 1 to 8 packets per second. The small differences between STAMP and other protocols are due to the false positives that the TMS could detect, which lead to a new route discovery process.

The effect of malicious nodes on the amount of control packets circulating in the network is shown in Fig. 6.12. Curves can be divided in two groups, depending on the drop percentage

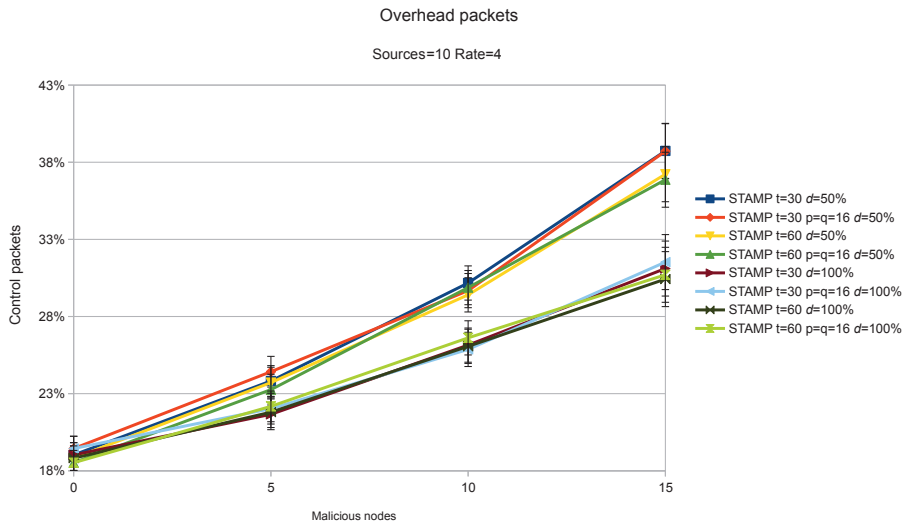


Fig. 6.12 Percentage of control packets vs the amount of malicious nodes in the network

of malicious nodes. There are two main reasons of increasing in overhead: the amount of malicious nodes and the percentage of packets they drop. The increase of misbehaving nodes leads to an increment of control packets, because a new route discovery process begins each time one of them is detected. Lower values of d make the detection harder, with malicious nodes having trust values closer to T_{th}, so they are rehabilitated, included in routes and detected again sooner (because their behavior does not change), so the amount of control packets increases. When the 30% of the nodes are malicious with d=50%, there is a small difference in curves in advantage of STAMP protocol with t=60, which has a smaller percentage of overhead with respect to the total amount of transmitted packets.

6.4.2 Detection accuracy

The analysis of the detection accuracy is done to understand if and in which way it is affected by the probabilistic monitoring. Our proposal is compared with the constant monitoring using various combinations of t, p and q parameters, so their effect on the performance can be evaluated.

Fig. 6.13 shows the percentage of true positives (detections made about malicious nodes). They seem to be weakly affected only by the t parameter, showing a similar trend for different values of q/p ratio in most cases. There is a difference of about 3% between the best and the worst case, therefore our proposal seems to not have any negative influence on the accuracy.

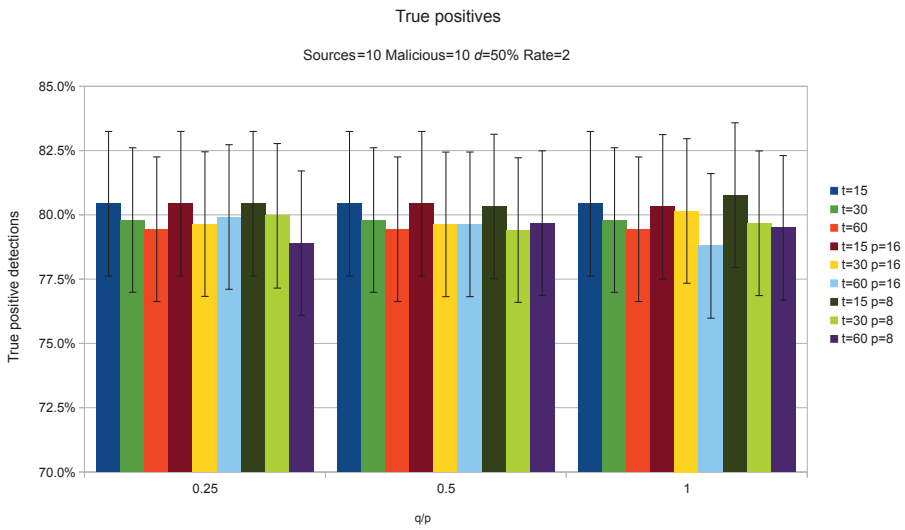


Fig. 6.13 Positive detections for different values of q/p

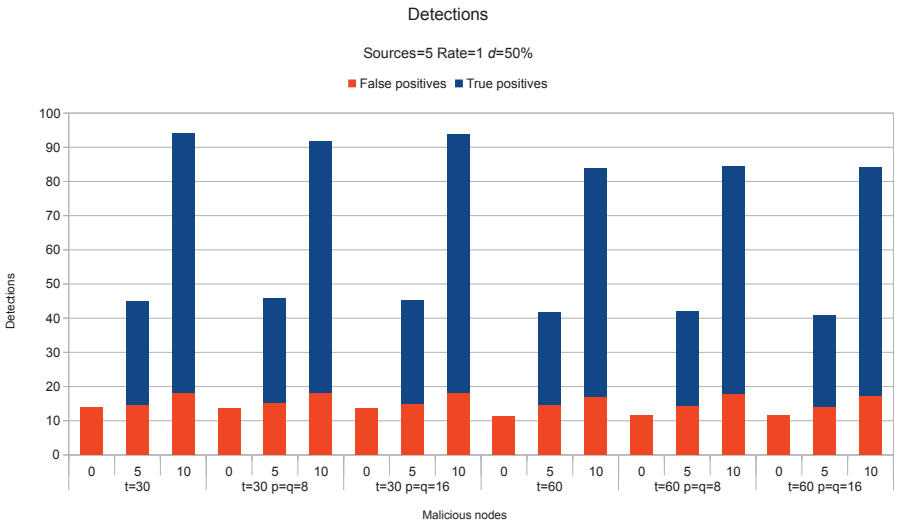


Fig. 6.14 Detections analysis with 5 source nodes and transmission rate of 1 packet per second for different amount of malicious nodes

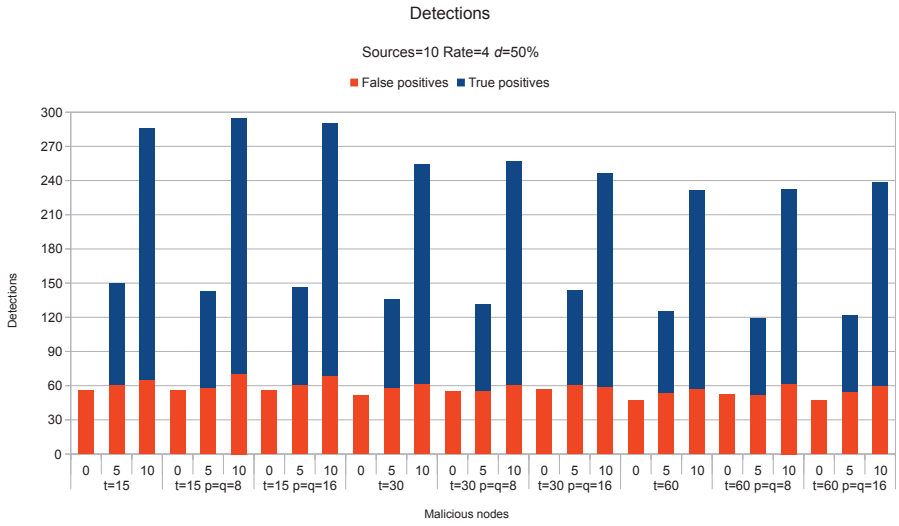


Fig. 6.15 Detections analysis with 10 source nodes and transmission rate of 4 packets per second for different amount of malicious nodes

The number of right and erroneous detections when 5 nodes receive packets from the same amount of source nodes and each of them sends one packet per second is shown in Fig. 6.14. The same results with a rate of four packets per second are shown in Fig. 6.15. The total amount of detections increases when more malicious nodes are in the network. False positive detections slightly increase proportionally with the amount of malicious nodes. They are generally higher with more source nodes and higher rates. Their absolute values are almost the same regardless of the number of attacker nodes, while true detections have a steep increase when the network is under the attack of more nodes. False positive detections are due to collisions and loss of links between moving nodes, so they cannot be avoided in MANET scenario. Using our proposal seems neither to improve or worsen the accuracy, the only difference brought by the parameters is on the total detections, which increase between the 7% and the 27% for smaller values of t.

6.4.3 Energy consumption analysis

The main aim of probabilistic monitoring is the reduction of the energy needed to realize a secure and trustworthy communication among nodes. The energy consumed to receive packets in promiscuous mode is analyzed, which is the way in which the direct trust of a node is computed. Furthermore, the analysis of the energy consumption due to cryptography with respect to the overall consumption was done.

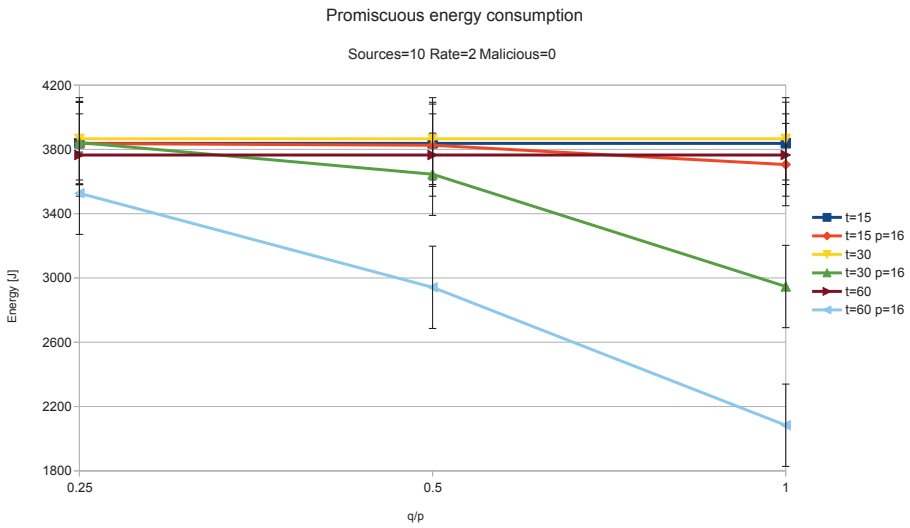


Fig. 6.16 Promiscuous energy consumption without malicious nodes vs q/p ratio

Promiscuous mode

The way in which the monitoring of transmitted packets is done requires the activation of the promiscuous mode of the wireless interface. Therefore, a node will overhear the data transmitted by its neighbor nodes to other nodes, which is the method used to observe the behavior of the neighbors. This operation requires more energy because the amount of received packet increases. Evaluating the energy consumption due to the reception of packets in promiscuous mode can give an idea of the consumption needed by the TMS.

In Fig. 6.16 the analysis of the energy consumption due to the promiscuous mode is shown. The results concern a network composed of fair nodes only. For constant monitoring, the cost is very similar, with a slight decrease when t value is higher. When probabilistic monitoring is used, better results are obtained when q/p ratio is higher. The best combination of parameters in terms of energy consumption is t=60 and p=q=16. Having higher t values increases the amount of observations taken into account by Eq. (4.2), so a fair node will keep higher trust values for a longer time. PMF shape is defined by p and q parameters, as shown in Fig. 6.1; having the q/p ratio equal to 1 allows to make less monitoring operations at lower trust values, while when $q/p < 1$ the trust value needed to reduce the PMF value is higher. The combination of these two effects allows to save more energy. For lower values of t, the amount of observations taken into account is lower, so the trust value tends to have values near T_{th}, obtaining a very small effect on the energy consumption, which is almost the same of the constant monitoring.

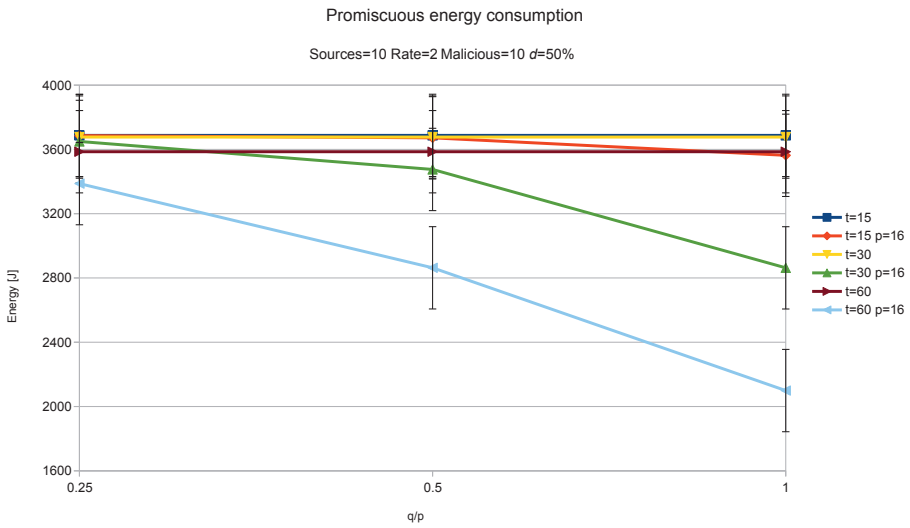


Fig. 6.17 Promiscuous energy consumption with 10 malicious nodes (d=50%) vs q/p ratio

The analysis of the promiscuous mode energy consumption in presence of malicious nodes is shown in Fig. 6.17. The trend is the same of the network without malicious nodes, the only difference is due to the slightly lower energy consumed. This little improvement is due to the lower amount of packets received in this mode, because malicious nodes do not forward a percentage of packets they receive. For higher d values, the trend is the same with a little more energy saved.

Cryptography

SAODV uses a mechanism of signatures and hash chains to ensure protocol packets against attacks. The attacker model defined in Section 4.3 is effective against SAODV, so the purpose of STAMP is protecting the network against this type of attack without consuming too much energy. The proposal is based on SAODV to have a protection against barely detectable attacks with the adopted TMS. We analyzed the cryptography energy consumption in two different methods:

1. only the costs to sign and verify a packet, and to apply the hash function to variable fields in protocol packets are taken into account;
2. the energy consumption due to the increased size of control packets including the extension fields introduced by cryptography is included in cryptography cost.

The second method is computed by analyzing the average cost respect to idle mode for each transmitted and received overhead packet for protocols with and without cryptography,

Table 6.7 Average Cost of Each Control Packet

Protocol	Tx Cost	Rx Cost
AODV	864.6 mJ	217.1 mJ
SAODV	1404 mJ	359.2 mJ
STAMP	1403.6 mJ	358.8 mJ

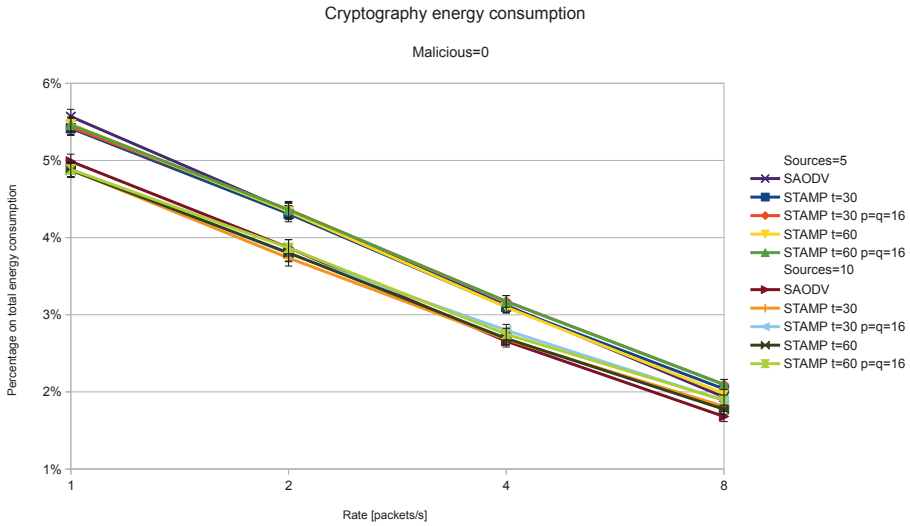


Fig. 6.18 Cryptography cost without malicious nodes for increasing data rates

so the difference for each packet can be included in cryptography cost instead of wireless transmission cost. The average cost for each transmitted and received control packet for different protocols is shown in Table 6.7.

The cryptography energy consumption without malicious nodes in the network is shown in Fig. 6.18. Two groups of curves can be noted. The percentage of consumption is almost the same with the same number of source nodes. There is a difference of about 0.5% between the groups, which decreases for higher transmission rates. The consumption has a smaller effect on the total energy consumption. The amount of sent data packets increases with more source nodes and at higher transmission rate. The higher percentage of cryptography cost respect to the total consumption is reached for the smallest simulated values of sink nodes and transmission rate.

The analysis of the cryptography cost in presence of malicious nodes is shown in Fig. 6.19. Using STAMP, we can note again two groups of curves, depending on the number of source nodes. Energy consumption is higher on average with respect to the network with fair nodes only. The behavior of SAODV protocol is slightly different. Its consumption is higher than STAMP when the transmission rate is one packet per second, while the consumption is

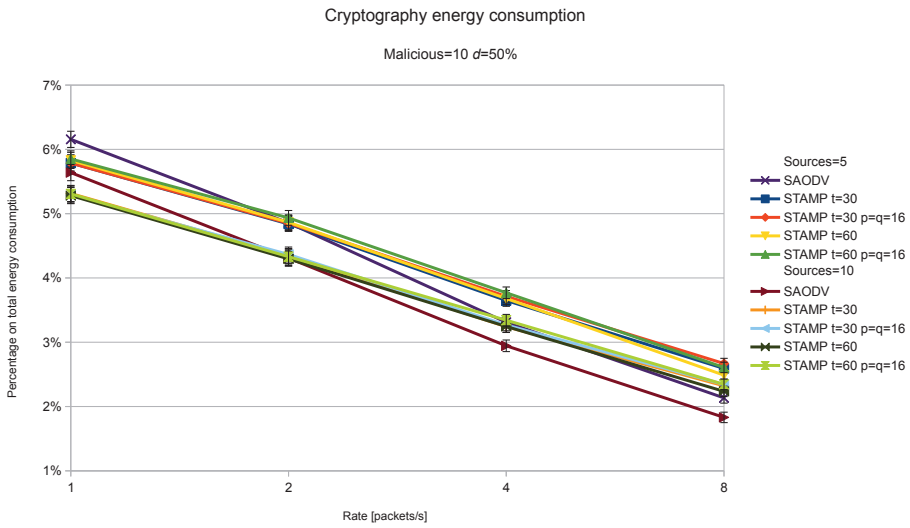


Fig. 6.19 Cryptography energy consumption with 10 malicious nodes for increasing data rates

the lowest for a rate of 8 packets per second. This difference could depend on the packets dropped by malicious nodes, against which SAODV has no protection.

When we include in the cryptography energy consumption the cost due to the increased control packets size, results highly change, as shown in the following. In Fig. 6.20 we show the energy consumption with a network composed of fair nodes only. The most important difference with respect to the results shown before is the percentage of consumption respect to the overall. When taking into account only the cryptographic operations, the percentage was always under the 7% of the total consumption. Now the percentage is almost the 30% in the worst case. It means that the most of the cryptography cost in routing protocols for wireless ad-hoc networks is due to the increased size of the overhead. All the curves follow the same trend, just SAODV seems to have a less percentage of cryptography cost for higher transmission rates. Transmitting data packets at higher rates leads to a decrease of the percentage of energy consumption due to cryptography. In Fig. 6.21 the cryptography consumption including the variation in overhead cost is shown. The difference between STAMP and SAODV is now wider. The maximum percentage reached, with a rate of one packet per second, is higher because the amount of control packets for STAMP increases due to the need of excluding malicious nodes from the routes. While for SAODV, the difference is due to the lower amount of data packets circulating in the network, because malicious nodes drop a percentage of received packets. Increasing the transmission rate of the data packets leads to a decrease in the percentage of cryptography cost. All the combinations of

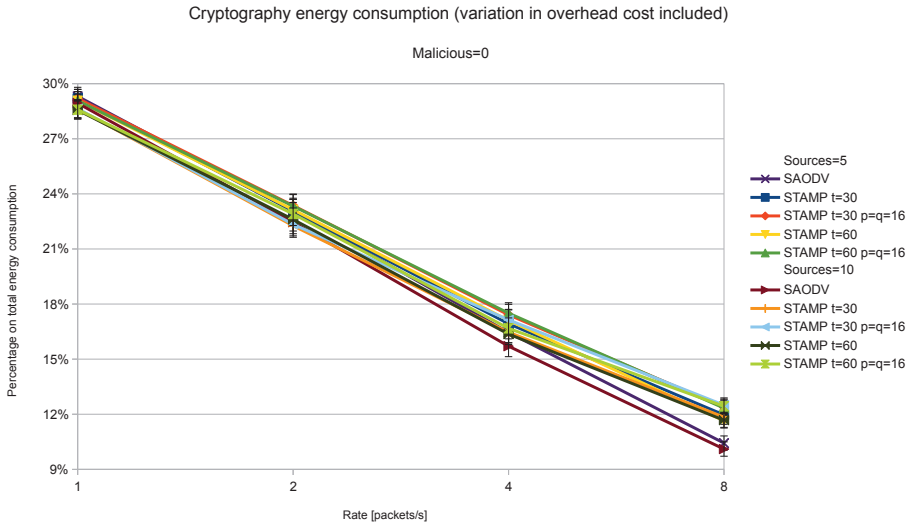


Fig. 6.20 Cryptography energy consumption including variation in overhead cost without malicious nodes for increasing data rates

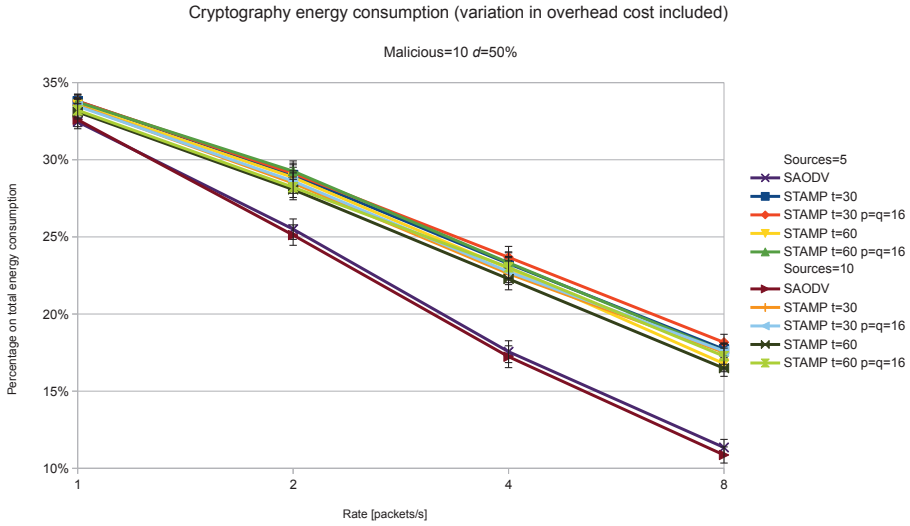


Fig. 6.21 Cryptography energy consumption including variation in overhead cost with 10 malicious nodes for increasing data rates

parameters for STAMP protocol lead to almost the same result, so probabilistic monitoring seems to have no effect on these results.

Chapter 7

Conclusions and Future Works

In this work, the effects of trust-based IDS in wireless ad-hoc networks were analyzed under the energy consumption perspective. The introduction of a TMS over SAODV protocol allowed increasing the network security, as seen in Chapter 4. The communication between the nodes is not disrupted if some nodes in the network drop maliciously packets that have to be forwarded, because the transmission will reach its destination by excluding the nodes that not behave correctly from the routes. The main difference of introducing a trust-based IDS in terms of consumption is due to the use of promiscuous mode exploited to detect nodes behaving maliciously, because a node receives all the data transmitted in its wireless range, and consequently it consumes more energy to receive and analyze them. In wireless ad-hoc networks composed of resource-constrained nodes, with the network prone to attacks, there is the need of finding a right trade-off between these two contrasting requirements.

Securing routing protocols in wireless ad-hoc networks is an expensive operation in terms of energy consumption. Both cryptography and trust management require large amount of energy, but they are unavoidable in these networks due to their characteristics. In order to efficiently apply an IDS on these networks, two different approaches were proposed. The distributed IDS presented in Chapter 5 allows enhancing security in mobile and collaborative networks. The way in which the monitoring is distributed among trusted neighbors enables the reduction of energy-consumption by reducing monitoring activities. CNs are chosen by taking into account their trustworthiness and the stability of the links connecting to them, computed over a statistical basis. The proposal effectiveness is validated through combining it with AODV routing protocol. Simulation results obtained using NS-3 show its effectiveness at both keeping low the amount of consumed energy and maintaining a high level of accuracy in detection. The proposal could be improved by extending this detection scheme to other

type of malicious behaviors. Moreover, the comparison with other similar proposal could help in understanding better the performance of this distributed IDS.

Concerning the probabilistic monitoring, it enables the use of an IDS in environments characterized by limited amount of available energy. The proposed approach could also improve the performance of the IDS, with less time needed to detect misbehaviors, although the main aim of PMF concerns the energy saving. The detection accuracy is affected by the probabilistic monitoring in a positive manner, achieving a little improvement against the constant monitoring. The defined constraints allow maintaining an accurate detection also when the nodes misbehavior is hardly noticeable. Under the energy consumption point of view, the results show a reduction of the consumption of 70% in the best case.

The analysis in a more complex network was possible by implementing the probabilistic monitoring and the TMS over NS-3, a well-known simulator commonly used by scientific community. Using the proposed TMS improves performance when a part of nodes participating in the network does not behave fairly. The increase of energy consumption is the price of securing the network. Results show few differences between constant and probabilistic monitoring under each analyzed point of view except the energy consumption, where the STAMP protocol achieves better performances. Reducing the energy consumption enables the usage of the proposed scheme to environments characterized by battery-constrained devices, as wireless ad-hoc networks. Analyzing cryptographic costs, their effect on the energy consumption is mainly due to the change in overhead packets. Transmitting and receiving control packets is more expensive because digital signatures and hash chains increase their size, so the energy consumed by the wireless interface increases.

Future developments on this proposal could be addressed in improving TMS performance by using a routing protocol that exploits backup routes or allows the reparation of a broken route by an intermediate node. The proposed approach marks a route as broken each time a malicious node is detected on that route, therefore implementing the TMS in a protocol that provides one of these enhancements will improve the energy saving. Local reparation of links and backup routes will decrease the overhead generated by reducing the amount of triggered route discovery processes. The proposed IDS could be further improved by designing, including and analyzing a protection mechanism against other threats, as DoS, DDoS and other distributed attacks. Another enhancement should concern the dynamic adaptation of TMS and PMF parameters to the network status. If no malicious agents are detected for enough time, ρ_t, p and q parameters could be changed according to it, diminishing the control over the transmission. When nodes detect a deterioration of the networks due to

misbehaving nodes, parameters value should be changed again to reinforce the monitoring and thus detecting and isolating malicious agents.

References

[1] M. Conti and S. Giordano, "Mobile ad hoc networking: milestones, challenges, and new research directions," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 85–96, January 2014.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393 – 422, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128601003024

[3] Y. Wang and F. Li, *Vehicular Ad Hoc Networks*. London: Springer London, 2009, pp. 503–525. [Online]. Available: http://dx.doi.org/10.1007/978-1-84800-328-6_20

[4] P. Fazio, F. De Rango, and A. Lupia, "Vehicular networks and road safety: An application for emergency/danger situations management using the WAVE/802.11p standard," *Advances in Electrical and Electronic Engineering*, vol. 11, no. 5, p. 357, 2013.

[5] M. Fogué, P. Garrido, F. J. Martinez, J. C. Cano, C. T. Calafate, and P. Manzoni, "Automatic accident detection: Assistance through communication technologies and vehicles," *IEEE Vehicular Technology Magazine*, vol. 7, no. 3, pp. 90–100, Sept 2012.

[6] İlker Bekmezci, O. K. Sahingoz, and Şamil Temel, "Flying ad-hoc networks (FANETs): A survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254 – 1270, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870512002193

[7] "IEEE standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications," *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pp. 1–2793, March 2012.

[8] "IEEE standard for information technology– telecommunications and information exchange between systemslocal and metropolitan area networks– specific requirements–part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications–amendment 4: Enhancements for very high throughput for operation in bands below 6 GHz." *IEEE Std 802.11ac-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa-2012, and IEEE Std 802.11ad-2012)*, pp. 1–425, Dec 2013.

[9] "IEEE standard for information technology–telecommunications and information exchange between systems–local and metropolitan area networks–specific

requirements-part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 3: Enhancements for very high throughput in the 60 GHz band," *IEEE Std 802.11ad-2012 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012 and IEEE Std 802.11aa-2012)*, pp. 1–628, Dec 2012.

[10] "IEEE standard for information technology - telecommunications and information exchange between systems - local and metropolitan area networks - specific requirements - part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 5: Television white spaces (TVWS) operation," *IEEE Std 802.11af-2013 (Amendment to IEEE Std 802.11-2012, as amended by IEEE Std 802.11ae-2012, IEEE Std 802.11aa-2012, IEEE Std 802.11ad-2012, and IEEE Std 802.11ac-2013)*, pp. 1–198, Feb 2014.

[11] E. Khorov, A. Lyakhov, A. Krotov, and A. Guschin, "A survey on IEEE 802.11ah: An enabling networking technology for smart cities," *Computer Communications*, vol. 58, pp. 53 – 69, 2015, special Issue on Networking and Communications for Smart Cities. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0140366414002989

[12] M. Dideles, "Bluetooth: A technical overview," *Crossroads*, vol. 9, no. 4, pp. 11–18, Jun. 2003. [Online]. Available: http://doi.acm.org/10.1145/904080.904083

[13] K. Townsend, C. Cufí, Akiba, and R. Davidson, *Getting Started with Bluetooth Low Energy: Tools and Techniques for Low-Power Networking*. O'Reilly Media, 2014.

[14] S. Farahani, *ZigBee Wireless Networks and Transceivers*. Newnes, 2011.

[15] S. Sesia, I. Toufik, and M. Baker, *LTE - The UMTS Long Term Evolution: From Theory to Practice*. Wiley, 2011.

[16] N. Bhushan, J. Li, D. Malladi, R. Gilmore, D. Brenner, A. Damnjanovic, R. T. Sukhavasi, C. Patel, and S. Geirhofer, "Network densification: the dominant theme for wireless evolution into 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 82–89, February 2014.

[17] E. M. Royer and C.-K. Toh, "A review of current routing protocols for ad hoc mobile wireless networks," *IEEE Personal Communications*, vol. 6, no. 2, pp. 46–55, Apr 1999.

[18] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc on-demand distance vector (AODV) routing," Internet Requests for Comments, RFC Editor, RFC 3561, July 2003. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3561.txt

[19] C. Perkins, S. Ratliff, J. Dowdell, L. Steenbrink, and V. Mercieca, "Ad hoc on-demand distance vector version 2 (AODVv2) routing," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-manet-aodvv2-16, May 2016. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-manet-aodvv2-16.txt

[20] T. Clausen and P. Jacquet, "Optimized link state routing protocol (OLSR)," Internet Requests for Comments, RFC Editor, RFC 3626, October 2003. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3626.txt

[21] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg, "The optimized link state routing protocol version 2," Internet Requests for Comments, RFC Editor, RFC 7181, April 2014. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7181.txt

[22] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," in *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, ser. SIGCOMM '94. New York, NY, USA: ACM, 1994, pp. 234–244. [Online]. Available: http://doi.acm.org/10.1145/190314.190336

[23] D. Johnson, Y. Hu, and D. Maltz, "The dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4," Internet Requests for Comments, RFC Editor, RFC 4728, February 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4728.txt

[24] M. G. Zapata and N. Asokan, "Securing ad hoc routing protocols," in *Proceedings of the 1st ACM Workshop on Wireless Security*, ser. WiSE '02. New York, NY, USA: ACM, 2002, pp. 1–10. [Online]. Available: http://doi.acm.org/10.1145/570681.570682

[25] B. Wu, J. Chen, J. Wu, and M. Cardei, *A Survey of Attacks and Countermeasures in Mobile Ad Hoc Networks*. Boston, MA: Springer US, 2007, pp. 103–135. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-33112-6_5

[26] A. Nadeem and M. P. Howarth, "A survey of MANET intrusion detection and prevention approaches for network layer attacks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 2027–2045, Fourth 2013.

[27] F.-H. Tseng, L.-D. Chou, and H.-C. Chao, "A survey of black hole attacks in wireless mobile ad hoc networks," *Human-centric Computing and Information Sciences*, vol. 1, no. 1, p. 4, 2011. [Online]. Available: http://dx.doi.org/10.1186/2192-1962-1-4

[28] A. Lupia and F. De Rango, "Evaluation of the energy consumption introduced by a trust management scheme on mobile ad-hoc networks," *Journal of Networks*, vol. 10, no. 4, 2015.

[29] J. Sen, M. G. Chandra, S. G. Harihara, H. Reddy, and P. Balamuralidhar, "A mechanism for detection of gray hole attack in mobile ad hoc networks," in *2007 6th International Conference on Information, Communications Signal Processing*, Dec 2007, pp. 1–5.

[30] G. Usha and S. Bose, "Impact of gray hole attack on adhoc networks," in *Information Communication and Embedded Systems (ICICES), 2013 International Conference on*, Feb 2013, pp. 404–409.

[31] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Wormhole attacks in wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 370–380, Feb 2006.

[32] ——, "Rushing attacks and defense in wireless ad hoc network routing protocols," in *Proceedings of the 2Nd ACM Workshop on Wireless Security*, ser. WiSe '03. New York, NY, USA: ACM, 2003, pp. 30–40. [Online]. Available: http://doi.acm.org/10.1145/941311.941317

[33] J. Newsome, E. Shi, D. Song, and A. Perrig, "The sybil attack in sensor networks: Analysis & defenses," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, ser. IPSN '04. New York, NY, USA: ACM, 2004, pp. 259–268. [Online]. Available: http://doi.acm.org/10.1145/984622.984660

[34] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An on-demand secure routing protocol resilient to byzantine failures," in *Proceedings of the 1st ACM Workshop on Wireless Security*, ser. WiSE '02. New York, NY, USA: ACM, 2002, pp. 21–30. [Online]. Available: http://doi.acm.org/10.1145/570681.570684

[35] J. Milliken, *Introduction to Wireless Intrusion Detection Systems*. Auerbach Publications, Dec 2013, pp. 335–360, 0. [Online]. Available: http://dx.doi.org/10.1201/b16390-19

[36] I. Butun, S. D. Morgera, and R. Sankar, "A survey of intrusion detection systems in wireless sensor networks," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 266–282, First 2014.

[37] T. Anantvalee and J. Wu, *A Survey on Intrusion Detection in Mobile Ad Hoc Networks*. Boston, MA: Springer US, 2007, pp. 159–180. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-33112-6_7

[38] K. Govindan and P. Mohapatra, "Trust computations and trust dynamics in mobile adhoc networks: A survey," *Communications Surveys Tutorials, IEEE*, vol. 14, no. 2, pp. 279–298, Second 2012.

[39] T. Zahariadis, H. C. Leligou, P. Trakadas, and S. Voliotis, "Trust management in wireless sensor networks," *European Transactions on Telecommunications*, vol. 21, no. 4, pp. 386–395, 2010. [Online]. Available: http://dx.doi.org/10.1002/ett.1413

[40] W. Li, W. Meng, L.-F. Kwok, and H. H. IP, "Enhancing collaborative intrusion detection networks against insider attacks using supervised intrusion sensitivity-based trust management model," *Journal of Network and Computer Applications*, vol. 77, pp. 135 – 145, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804516302211

[41] S. Tan, X. Li, and Q. Dong, "Trust based routing mechanism for securing OSLR-based MANET," *Ad Hoc Networks*, vol. 30, pp. 84–98, 2015.

[42] J.-H. Cho, A. Swami, and I.-R. Chen, "Modeling and analysis of trust management with trust chain optimization in mobile ad hoc networks," *Journal of Network and Computer Applications*, vol. 35, no. 3, pp. 1001–1012, 2012, special Issue on Trusted Computing and Communications.

[43] ——, "A survey on trust management for mobile ad hoc networks," *Communications Surveys Tutorials, IEEE*, vol. 13, no. 4, pp. 562–583, Fourth 2011.

[44] I.-R. Chen, J. Guo, F. Bao, and J.-H. Cho, "Trust management in mobile ad hoc networks for bias minimization and application performance maximization," *Ad Hoc Networks*, vol. 19, pp. 59 – 74, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870514000419

[45] H. Zhu, S. Du, Z. Gao, M. Dong, and Z. Cao, "A probabilistic misbehavior detection scheme toward efficient trust establishment in delay-tolerant networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 22–32, Jan 2014.

[46] H. Xia, Z. Jia, X. Li, L. Ju, and E. H.-M. Sha, "Trust prediction and trust-based source routing in mobile ad hoc networks," *Ad Hoc Networks*, vol. 11, no. 7, pp. 2096 – 2114, 2013, theory, Algorithms and Applications of Wireless Networked RoboticsRecent Advances in Vehicular Communications and Networking. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870512000261

[47] R. Feng, S. Che, X. Wang, and N. Yu, "A credible routing based on a novel trust mechanism in ad hoc networks," *International Journal of Distributed Sensor Networks*, vol. 9, no. 4, p. 652051, 2013. [Online]. Available: http://dx.doi.org/10.1155/2013/652051

[48] F. De Rango, "Trust-based SAODV protocol with intrusion detection, trust management and incentive cooperation in MANETs," *International Journal of Interdisciplinary Telecommunications and Networking (IJITN)*, vol. 1, no. 4, pp. 54–70, 2009. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jitn.2009092804

[49] F. De Rango and S. Marano, "Trust-based saodv protocol with intrusion detection and incentive cooperation in manet," in *Proceedings of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, ser. IWCMC '09. New York, NY, USA: ACM, 2009, pp. 1443–1448. [Online]. Available: http://doi.acm.org/10.1145/1582379.1582695

[50] F. De Rango, "Improving SAODV protocol with trust levels management, idm and incentive cooperation in MANET," in *2009 Wireless Telecommunications Symposium*, April 2009, pp. 1–8.

[51] S. Sarkar and R. Datta, "A trust based protocol for energy-efficient routing in self-organized manets," in *2012 Annual IEEE India Conference (INDICON)*, Dec 2012, pp. 1084–1089.

[52] M. Virendra, M. Jadliwala, M. Chandrasekaran, and S. Upadhyaya, "Quantifying trust in mobile ad-hoc networks," in *International Conference on Integration of Knowledge Intensive Multi-Agent Systems, 2005.*, April 2005, pp. 65–70.

[53] T. AlSkaif, M. G. Zapata, and B. Bellalta, "Game theory for energy efficiency in wireless sensor networks: Latest trends," *Journal of Network and Computer Applications*, vol. 54, pp. 33 – 61, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804515000806

[54] Z. Chen, C. Qiao, Y. Qiu, L. Xu, and W. Wu, "Dynamics stability in wireless sensor networks active defense model," *Journal of Computer and System Sciences*, vol. 80, no. 8, pp. 1534 – 1548, 2014, special Issue on Theory and Applications in Parallel and Distributed Computing Systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0022000014000646

[55] J. Duan, D. Gao, D. Yang, C. H. Foh, and H. H. Chen, "An energy-aware trust derivation scheme with game theoretic approach in wireless sensor networks for iot applications," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 58–69, Feb 2014.

[56] J. Liu, G. Yue, S. Shen, H. Shang, and H. Li, "A game-theoretic response strategy for coordinator attack in wireless sensor networks," *The Scientific World Journal*, 2014. [Online]. Available: http://dx.doi.org/10.1155/2014/950618

[57] K. Khalil, Z. Qian, P. Yu, S. Krishnamurthy, and A. Swami, "Optimal monitor placement for detection of persistent threats," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[58] J. t. Wang, Z. g. Chen, and X. h. Deng, "A trustworthy energy-efficient routing algorithm based on game-theory for wsn," in *IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2009)*, Dec 2009, pp. 192–196.

[59] M. Asadi, C. Zimmerman, and A. Agah, "A game-theoretic approach to security and power conservation in wireless sensor networks," *International Journal of Network Security*, vol. 15, no. 1, pp. 50–58, Jan 2013.

[60] S. Shamshirband, A. Patel, N. B. Anuar, M. L. M. Kiah, and A. Abraham, "Cooperative game theoretic approach using fuzzy q-learning for detecting and preventing intrusions in wireless sensor networks," *Engineering Applications of Artificial Intelligence*, vol. 32, pp. 228 – 241, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0952197614000311

[61] A. Saeed, A. Ahmadinia, A. Javed, and H. Larijani, "Random neural network based intelligent intrusion detection for wireless sensor networks," *Procedia Computer Science*, vol. 80, pp. 2372 – 2376, 2016, international Conference on Computational Science 2016, {ICCS} 2016, 6-8 June 2016, San Diego, California, {USA}. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050916309371

[62] Y. Maleh, A. Ezzati, Y. Qasmaoui, and M. Mbida, "A global hybrid intrusion detection system for wireless sensor networks," *Procedia Computer Science*, vol. 52, pp. 1047 – 1052, 2015, the 6th International Conference on Ambient Systems, Networks and Technologies (ANT-2015), the 5th International Conference on Sustainable Energy Information Technology (SEIT-2015). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915009084

[63] M. G. Zapata, "Secure ad hoc on-demand distance vector (saodv) routing," Working Draft, IETF Secretariat, Internet-Draft draft-guerrero-manet-saodv-06, September 2006. [Online]. Available: http://www.ietf.org/internet-drafts/draft-guerrero-manet-saodv-06.txt

[64] J. von Mulert, I. Welch, and W. K. Seah, "Security threats and solutions in MANETs: A case study using AODV and SAODV," *Journal of Network and Computer Applications*, vol. 35, no. 4, pp. 1249 – 1259, 2012, intelligent Algorithms for Data-Centric Sensor Networks.

[65] Y.-C. Hu, D. B. Johnson, and A. Perrig, "Sead: secure efficient distance vector routing for mobile wireless ad hoc networks," *Ad Hoc Networks*, vol. 1, no. 1, pp. 175 – 192, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870503000192

[66] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," *Wirel. Netw.*, vol. 11, no. 1-2, pp. 21–38, Jan. 2005. [Online]. Available: http://dx.doi.org/10.1007/s11276-004-4744-y

[67] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *In Network and Distributed System Security Symposium, NDSS '01*, 2001, pp. 35–46.

[68] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer, "A secure routing protocol for ad hoc networks," in *10th IEEE International Conference on Network Protocols, 2002. Proceedings.*, Nov 2002, pp. 78–87.

[69] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: Security protocols for sensor networks," *Wirel. Netw.*, vol. 8, no. 5, pp. 521–534, Sep. 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1016598314198

[70] Z. Wan, K. Ren, and M. Gu, "Usor: An unobservable secure on-demand routing protocol for mobile ad hoc networks," *IEEE Transactions on Wireless Communications*, vol. 11, no. 5, pp. 1922–1932, May 2012.

[71] L. Sellami, D. Idoughi, A. Baadache, and P. Tiako, "A novel detection intrusion approach for ubiquitous and pervasive environments," *Procedia Computer Science*, vol. 94, pp. 429 – 434, 2016, the 11th International Conference on Future Networks and Communications (FNC 2016) / The 13th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2016) / Affiliated Workshops. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050916318166

[72] E. Hernandez-Orallo, M. Serrat Olmos, J.-C. Cano, C. Calafate, and P. Manzoni, "CoCoWa: A collaborative contact-based watchdog for detecting selfish nodes," *Mobile Computing, IEEE Transactions on*, vol. 14, no. 6, pp. 1162–1175, June 2015.

[73] Y. Sun, W. Yu, Z. Han, and K. Liu, "Information theoretic framework of trust modeling and evaluation for ad hoc networks," *Selected Areas in Communications, IEEE Journal on*, vol. 24, no. 2, pp. 305–317, Feb 2006.

[74] L. M. Feeney, "An energy consumption model for performance analysis of routing protocols for mobile ad hoc networks," *Mobile Networks and Applications*, vol. 6, no. 3, pp. 239–249, Jun. 2001.

[75] D. Halperin, B. Greenstein, A. Sheth, and D. Wetherall, "Demystifying 802.11n power consumption," in *Proceedings of the 2010 International Conference on Power Aware Computing and Systems*, ser. HotPower'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–. [Online]. Available: http://dl.acm.org/citation.cfm?id=1924920.1924928

[76] N. R. Potlapally, S. Ravi, A. Raghunathan, and N. K. Jha, "A study of the energy consumption characteristics of cryptographic algorithms and security protocols," *IEEE Transactions on Mobile Computing*, vol. 5, no. 2, pp. 128–143, Feb 2006.

[77] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Kluwer Academic Publishers, 1996, pp. 153–181.

[78] A. Lupia, C. A. Kerrache, F. De Rango, C. T. Calafate, J.-C. Cano, and P. Manzoni, "TEEM: Trust-based energy-efficient distributed monitoring for mobile ad-hoc networks," in *2017 IFIP Wireless Days (WD)*, Mar 2017.

[79] F. De Rango, F. Guerriero, and P. Fazio, "Link-stability and energy aware routing protocol in distributed wireless networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 4, pp. 713–726, April 2012.

[80] C. A. Kerrache, N. Lagraa, C. T. Calafate, and A. Lakas, "TROUVE: A trusted routing protocol for urban vehicular environments," in *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, Oct 2015, pp. 260–267.

[81] A. Lupia and F. De Rango, "Trust management using probabilistic energy-aware monitoring for intrusion detection in mobile ad-hoc networks," in *2016 Wireless Telecommunications Symposium (WTS)*, April 2016, pp. 1–6.

[82] ——, "A probabilistic energy-efficient approach for monitoring and detecting malicious/selfish nodes in mobile ad-hoc networks," in *2016 IEEE Wireless Communications and Networking Conference*, April 2016, pp. 1–6.

[83] R. Chattamvelli and R. Shanmugam, "Continuous distributions," in *Statistics for Scientists and Engineers*. John Wiley & Sons, 2015, pp. 255–332.

[84] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Güneş, and J. Gross, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 15–34.

[85] Boost C++ Libraries. (visited on Feb. 15, 2017). [Online]. Available: http://www.boost.org/

[86] Crypto++ Library. (visited on Feb. 15, 2017). [Online]. Available: http://www.cryptopp.com/

Appendix A

NS-3 Code

The full code of the *tms* module used for simulation in NS-3.26 is illustrated in this section. For *aodv* module, the result of the diff command used for the comparison with the code included in NS-3.26 is shown.

A.1 Module *tms*

The module is composed of the classes listed in this section. After the creation of the directory "tms" in the source directory of NS-3, two more directories need to be created: "model" and "helper". The following files have to be put inside the "model" folder:

- *trust-manager.h*;
- *trust-manager.cc*;
- *trust-pmf.h*;
- *trust-pmf.cc*;
- *trust-table.h*;
- *trust-table.cc*;

The files to put in "helper" directory are:

- *tms-helper.h*;
- *tms-helper.cc*.

The file *wscript* has to be placed in the main folder of the module, which is "tms".

trust-manager.h

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* Copyright (c) 2016 CULTURE TELELAB, DIMES - Universita' della Calabria
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
*
* Author: Andrea Lupia <alupia@dimes.unical.it>
*/

#ifndef TRUST_MANAGER_H
#define TRUST_MANAGER_H

#include <map>
#include <list>
#include "trust-table.h"
#include "trust-pmf.h"
#include "ns3/object.h"
#include "ns3/ptr.h"
#include "ns3/ipv4-l3-protocol.h"
#include "ns3/packet.h"
#include "ns3/wifi-net-device.h"
#include "ns3/ipv4-address.h"
#include "ns3/timer.h"
#include "ns3/random-variable-stream.h"
#include "ns3/wifi-mac-header.h"

namespace ns3 {

class Node;

namespace tms {

struct BufferEntry
{
Timer *m_nextHopWait;
uint64_t m_uid;
Ipv4Address m_address;

BufferEntry (uint64_t uid, Ipv4Address address) :
m_uid (uid), m_address (address)
{
m_nextHopWait = new Timer(Timer::REMOVE_ON_DESTROY);
}
};

class TrustManager : public Object
{
public:
static TypeId GetTypeId ();
TypeId GetInstanceTypeId () const;
TrustManager ();
virtual ~TrustManager ();
Ptr<TrustTable> GetTrustTable ();

bool IsProm () { return m_prom; }
protected:
virtual void DoInitialize (void);
private:
```

```
// void ManageTx (Ptr<const Packet> packet, Ptr<Ipv4> ipv4,  uint32_t interface);
void ManageTx (const Ipv4Header &header, Ptr<const Packet> packet, uint32_t interface);
// bool
void ManagePromiscRx (Ptr<NetDevice> device, Ptr<const Packet> packet, uint16_t protocol, const Address &from,
const Address &to, NetDevice::PacketType packetType);
void ManageTxError (WifiMacHeader const & hdr);

bool InsertInBuffer (uint64_t uid, Ipv4Address address);
bool EmptyBuffer ();
bool RemoveFromBuffer (uint64_t uid, Ipv4Address address);

Ipv4Address GetIpFromMac (Mac48Address address);
void NextHopWaitExpire (uint64_t uid, Ipv4Address address);

void SetProbabilisticMonitoring (bool f);
bool IsProbabilisticMonitoring () const;

void SetP (double p);
double GetP () const;

void SetQ (double q);
double GetQ () const;

void SetRho (double rho);
double GetRho () const;

void SetEpsilon (double epsilon);
double GetEpsilon () const;

void SetProm (bool prom) { m_prom = prom; }

Ptr<Node> m_node;
Ptr<Ipv4L3Protocol> m_ipv4;
Ptr<WifiNetDevice> m_device;

Ptr<UniformRandomVariable> m_urv;

std::map<uint64_t, BufferEntry> m_buffer;

Ptr<TrustTable> m_table;
Ptr<ProbabilisticMonitoring> m_pmf;

// Parameters
Time m_trustUpdateInterval;
Time m_nextHopWait;
bool m_probabilistic;
double m_p;
double m_q;
double m_rho;
double m_epsilon;

// Simulate promiscuous deactivation
bool m_prom;
};
}
}

#endif /* TRUST_MANAGER_H */
```

trust-manager.cc

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* Copyright (c) 2016 CULTURE TELELAB, DIMES - Universita' della Calabria
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
```

```
8  *
9  * This program is distributed in the hope that it will be useful,
10 * but WITHOUT ANY WARRANTY; without even the implied warranty of
11 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
12 * GNU General Public License for more details.
13 *
14 * You should have received a copy of the GNU General Public License
15 * along with this program; if not, write to the Free Software
16 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
17 *
18 * Author: Andrea Lupia <alupia@dimes.unical.it>
19 */
20
21 #include "trust-manager.h"
22 #include "ns3/log.h"
23 #include "ns3/pointer.h"
24 #include "ns3/boolean.h"
25 #include "ns3/double.h"
26 #include "ns3/simulator.h"
27 #include "ns3/node.h"
28 #include "ns3/config.h"
29 #include "ns3/packet.h"
30 #include "ns3/wifi-mac.h"
31 #include "ns3/node-list.h"
32 #include "ns3/timer.h"
33 #include "ns3/mac-low.h"
34
35 namespace ns3 {
36 namespace tms {
37
38 NS_LOG_COMPONENT_DEFINE ("TrustManager");
39 NS_OBJECT_ENSURE_REGISTERED (TrustManager);
40
41 TypeId
42 TrustManager::GetTypeId (void)
43 {
44 static TypeId tid = TypeId ("ns3::tms::TrustManager")
45   .SetParent<Object> ()
46   .SetGroupName ("Tms")
47   .AddConstructor<TrustManager> ()
48   .AddAttribute ("Node",
49     "Reference node.",
50     PointerValue (),
51     MakePointerAccessor (&TrustManager::m_node),
52     MakePointerChecker<Node> ())
53   .AddAttribute ("TrustUpdateInterval",
54     "Trust values update interval.",
55     TimeValue (Seconds (2)),
56     MakeTimeAccessor (&TrustManager::m_trustUpdateInterval),
57     MakeTimeChecker ())
58   .AddAttribute ("NextHopWait",
59     "Period of observation waiting until observation is negative.",
60     TimeValue (MilliSeconds (50)),
61     MakeTimeAccessor (&TrustManager::m_nextHopWait),
62     MakeTimeChecker ())
63   .AddAttribute ("Probabilistic",
64     "Indicates whether the probabilistic monitoring is enabled.",
65     BooleanValue (true),
66     MakeBooleanAccessor (&TrustManager::SetProbabilisticMonitoring,
67       &TrustManager::IsProbabilisticMonitoring),
68     MakeBooleanChecker ())
69   .AddAttribute ("BetaP",
70     "The \"p\" parameter of the Probabilistic Monitoring Function.",
71     DoubleValue (4.),
72     MakeDoubleAccessor (&TrustManager::SetP,
73       &TrustManager::GetP),
74     MakeDoubleChecker<double> (0.))
75   .AddAttribute ("BetaQ",
76     "The \"q\" parameter of the Probabilistic Monitoring Function.",
77     DoubleValue (4.),
78     MakeDoubleAccessor (&TrustManager::SetQ,
79       &TrustManager::GetQ),
```

```
80      MakeDoubleChecker<double> (0.))
81    .AddAttribute ("Rho",
82      "The remembering factor of the trust framework.",
83      DoubleValue (0.89),
84      MakeDoubleAccessor (&TrustManager::SetRho,
85        &TrustManager::GetRho),
86      MakeDoubleChecker<double> (0., 1.))
87    .AddAttribute ("Epsilon",
88      "The \"\\epsilon\" parameter of the probabilistic monitoring model",
89      DoubleValue (0.001),
90      MakeDoubleAccessor (&TrustManager::SetEpsilon,
91        &TrustManager::GetEpsilon),
92      MakeDoubleChecker<double> ())
93  ;
94  return tid;
95  }
96
97  TypeId
98  TrustManager::GetInstanceTypeId (void) const
99  {
100 return GetTypeId ();
101 }
102
103 TrustManager::TrustManager () :
104   m_trustUpdateInterval (Seconds (2)),
105   m_nextHopWait (MilliSeconds (50)),
106   m_probabilistic (true),
107   m_p (4.),
108   m_q (4.),
109   m_rho (0.89),
110   m_epsilon (0.001)
111 {
112 m_table = CreateObject<TrustTable> ();
113 }
114
115 TrustManager::~TrustManager ()
116 {
117 }
118
119 void
120 TrustManager::DoInitialize (void)
121 {
122 NS_LOG_FUNCTION (this << m_node->GetId ());
123 m_ipv4 = m_node->GetObject<Ipv4L3Protocol> ();
124 m_device = DynamicCast<WifiNetDevice> (m_node->GetDevice (0));
125 m_node->RegisterProtocolHandler (MakeCallback (&TrustManager::ManagePromiscRx, this),
126   Ipv4L3Protocol::PROT_NUMBER, m_device, true);
127 SetProm (false);
128 if (!m_ipv4->TraceConnectWithoutContext ("SendOutgoing",
129   MakeCallback (&TrustManager::ManageTx, this)))
130 {
131 NS_FATAL_ERROR ("trace fail");
132 }
133 if (!m_ipv4->TraceConnectWithoutContext ("UnicastForward",
134   MakeCallback (&TrustManager::ManageTx, this)))
135 {
136 NS_FATAL_ERROR ("trace fail");
137 }
138 // Allow neighbor manager use this interface for layer 2 feedback if possible
139 Ptr<WifiNetDevice> wifi = m_device->GetObject<WifiNetDevice> ();
140 Ptr<WifiMac> mac = wifi->GetMac ();
141 mac->TraceConnectWithoutContext ("TxErrHeader",
142   MakeCallback (&TrustManager::ManageTxError, this));
143 m_urv = CreateObject<UniformRandomVariable> ();
144 Type type = IsProbabilisticMonitoring () ? BETA : CONSTANT;
145 m_pmf = CreateObject<ProbabilisticMonitoring> (type);
146 if (m_pmf->IsProbabilistic ())
147 {
148 m_pmf->SetBetaParams (m_p, m_q);
149 m_pmf->CreateDistribution ();
150 }
151 m_table->SetRho (m_rho);
```

```
152 m_table->SetEpsilon (m_epsilon);
153 m_table->SetTrustUpdateInterval (m_trustUpdateInterval);
154 }
155
156 Ptr<TrustTable>
157 TrustManager::GetTrustTable ()
158 {
159 return m_table;
160 }
161
162 void
163 TrustManager::ManageTx (const Ipv4Header &header,
164   Ptr<const Packet> packet, uint32_t interface)
165 {
166 Ptr<Packet> p = packet->Copy();
167 Socket::SocketErrno socket_errno;
168 Ipv4Address dest = header.GetDestination ();
169 Ipv4Address nextHop = m_ipv4->GetRoutingProtocol()->
170 RouteOutput (p, header, m_device, socket_errno)->GetGateway ();
171 if(dest != nextHop && nextHop != Ipv4Address ("127.0.0.1")
172   && m_ipv4->IsUnicast (dest))
173 {
174 NS_LOG_DEBUG ("ManageTx: (destination " << dest <<
175   ", nexthop " << nextHop <<
176   ", packet " << p->GetUid () << ")");
177 // Probabilistic monitoring
178 double rnd = m_urv->GetValue (0, 1);
179 double probMon = m_pmf->GetMonitoringProbability (m_table->GetTrustValue (nextHop));
180 if (rnd < probMon)
181 {
182 NS_LOG_INFO ("The packet " << p->GetUid () << " will be monitored (" <<
183   rnd << " < " << probMon << ")");
184 InsertInBuffer (p->GetUid (), nextHop);
185 }
186 else
187 {
188 NS_LOG_INFO ("The packet " << p->GetUid () << " will NOT be monitored (" <<
189   rnd << " >= " << probMon << ")");
190 }
191 }
192 }
193
194 void
195 TrustManager::ManagePromiscRx (Ptr<NetDevice> device,
196   Ptr<const Packet> packet, uint16_t protocol, const Address &from,
197   const Address &to, NetDevice::PacketType packetType)
198 {
199 // Receive only IP packets and packets addressed to other hosts
200 if (packetType == NetDevice::PACKET_OTHERHOST && IsProm ())
201 {
202 Ptr<Packet> p = packet->Copy ();
203 Ipv4Header header;
204 p->RemoveHeader (header);
205
206 if (m_ipv4->IsUnicast (header.GetDestination()))
207 {
208 Ipv4Address promiscSource = GetIpFromMac (Mac48Address::ConvertFrom (from));
209 NS_LOG_DEBUG ("ManagePromiscRx: (source " << promiscSource <<
210   ", packet "<< p->GetUid () << ")");
211 if (RemoveFromBuffer (p->GetUid (), promiscSource))
212 {
213 NS_LOG_INFO ("Entry was found for packet " << p->GetUid () <<
214   " from source " << promiscSource);
215 // Add positive observation for promiscSource
216 m_table->AddObservation (promiscSource, true);
217 }
218 }
219 }
220 }
221
222 void
223 TrustManager::ManageTxError (WifiMacHeader const & hdr)
```

```
{
Ipv4Address address = GetIpFromMac (hdr.GetAddr1 ());
for (std::map<uint64_t, BufferEntry>::iterator i = m_buffer.begin ();
  i != m_buffer.end (); ++i)
{
if (i->second.m_address == address)
{
i->second.m_nextHopWait->Cancel ();
m_buffer.erase (i);
if (EmptyBuffer ())
{
SetProm (false);
}
}
}
}

bool
TrustManager::InsertInBuffer (uint64_t uid, Ipv4Address address)
{
BufferEntry entry (uid, address);
if (m_buffer.insert(std::make_pair (uid, entry)).second)
{
entry.m_nextHopWait->SetFunction (&TrustManager::NextHopWaitExpire, this);
entry.m_nextHopWait->SetArguments (uid, address);
entry.m_nextHopWait->SetDelay (m_nextHopWait);
entry.m_nextHopWait->Schedule ();
SetProm (true);
return true;
}
return false;
}

bool
TrustManager::EmptyBuffer ()
{
return m_buffer.empty ();
}

bool
TrustManager::RemoveFromBuffer (uint64_t uid, Ipv4Address address)
{
std::map<uint64_t, BufferEntry>::iterator i = m_buffer.find (uid);
if (i != m_buffer.end () && i->second.m_address == address)
{
i->second.m_nextHopWait->Cancel ();
m_buffer.erase (i);
if (EmptyBuffer ())
{
SetProm (false);
}
return true;
}
return false;
}

Ipv4Address
TrustManager::GetIpFromMac (Mac48Address address)
{
int32_t nNodes = NodeList::GetNNodes ();
for (int32_t i = 0; i < nNodes; ++i)
{
Ptr<Node> node = NodeList::GetNode (i);
Ptr<Ipv4> ipv4 = node->GetObject<Ipv4> ();
Ptr<NetDevice> netDevice = ipv4->GetNetDevice (1);
if (netDevice->GetAddress () == address)
{
return ipv4->GetAddress (1, 0).GetLocal ();
}
}
return 0;
}
```

```
296
297 void
298 TrustManager::NextHopWaitExpire (uint64_t uid, Ipv4Address address)
299 {
300 if (RemoveFromBuffer (uid, address))
301 {
302 NS_LOG_INFO ("Packet " << uid << " not sensed, negative observation for node "
303   << address);
304 // Add negative observation for address
305 m_table->AddObservation (address, false);
306 }
307 }
308
309 void
310 TrustManager::SetProbabilisticMonitoring (bool f)
311 {
312 m_probabilistic = f;
313 }
314
315 bool
316 TrustManager::IsProbabilisticMonitoring () const
317 {
318 return m_probabilistic;
319 }
320
321 void
322 TrustManager::SetP (double p)
323 {
324 m_p = p;
325 }
326
327 double
328 TrustManager::GetP () const
329 {
330 return m_p;
331 }
332
333 void
334 TrustManager::SetQ (double q)
335 {
336 m_q = q;
337 }
338
339 double
340 TrustManager::GetQ () const
341 {
342 return m_q;
343 }
344
345 void
346 TrustManager::SetRho (double rho)
347 {
348 m_rho = rho;
349 }
350
351 double
352 TrustManager::GetRho () const
353 {
354 return m_rho;
355 }
356
357 void
358 TrustManager::SetEpsilon (double epsilon)
359 {
360 m_epsilon = epsilon;
361 }
362
363 double
364 TrustManager::GetEpsilon () const
365 {
366 return m_epsilon;
367 }
```

```
}
}
```

trust-pmf.h

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* Copyright (c) 2016 CULTURE TELELAB, DIMES - Universita' della Calabria
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
*
* Author: Andrea Lupia <alupia@dimes.unical.it>
*/

#ifndef TRUST_PMF_H
#define TRUST_PMF_H

#include "ns3/object.h"
#include <boost/math/distributions/beta.hpp>

using boost::math::beta_distribution;

namespace ns3
{
namespace tms
{

enum Type
{
CONSTANT,
BETA
};

class ProbabilisticMonitoring : public Object
{
public:
static TypeId GetTypeId ();
TypeId GetInstanceTypeId () const;
ProbabilisticMonitoring (Type type);
void SetBetaParams (double p, double q);
void SetBetaP (double p);
void SetBetaQ (double q);
void CreateDistribution ();
double GetMonitoringProbability (double t);
bool IsProbabilistic ();
private:
Type m_type;
double m_p;
double m_q;
beta_distribution<> m_beta_pmf;
};

} /* namespace tms */
} /* namespace ns3 */

#endif /* TRUST_PMF_H */
```

trust-pmf.cc

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* Copyright (c) 2016 CULTURE TELELAB, DIMES - Universita' della Calabria
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
*
* Author: Andrea Lupia <alupia@dimes.unical.it>
*/

#include "trust-pmf.h"
#include "ns3/log.h"
#include "ns3/simulator.h"

using boost::math::beta_distribution;

namespace ns3
{
namespace tms
{

NS_LOG_COMPONENT_DEFINE ("ProbabilisticMonitoring");

NS_OBJECT_ENSURE_REGISTERED (ProbabilisticMonitoring);

TypeId
ProbabilisticMonitoring::GetTypeId (void)
{
static TypeId tid = TypeId ("ns3::ProbabilisticMonitoring")
.SetParent<Object> ()
.SetGroupName ("Tms")
;
return tid;
}

TypeId
ProbabilisticMonitoring::GetInstanceTypeId (void) const
{
return GetTypeId ();
}

ProbabilisticMonitoring::ProbabilisticMonitoring (Type type) :
m_type (type), m_p (1.), m_q (1.)
{

}

void
ProbabilisticMonitoring::SetBetaParams (double p, double q)
{
m_p = p;
m_q = q;
}

void
ProbabilisticMonitoring::SetBetaP (double p)
{
m_p = p;
}
```

```
70
71  void
72  ProbabilisticMonitoring::SetBetaQ (double q)
73  {
74  m_q = q;
75  }
76
77  void
78  ProbabilisticMonitoring::CreateDistribution ()
79  {
80  m_beta_pmf = beta_distribution<> (m_p, m_q);
81  }
82
83  double
84  ProbabilisticMonitoring::GetMonitoringProbability (double t)
85  {
86  double ans = 1.;
87  if (m_type != CONSTANT && t > 0)
88  {
89  ans -= boost::math::cdf (m_beta_pmf, t);
90  }
91  return ans;
92  }
93
94  bool
95  ProbabilisticMonitoring::IsProbabilistic ()
96  {
97  return m_type == BETA;
98  }
99
100 } /* namespace tms */
101 } /* namespace ns3 */
```

trust-table.h

```
1   /* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
2   /*
3   * Copyright (c) 2016 CULTURE TELELAB, DIMES - Universita' della Calabria
4   *
5   * This program is free software; you can redistribute it and/or modify
6   * it under the terms of the GNU General Public License version 2 as
7   * published by the Free Software Foundation;
8   *
9   * This program is distributed in the hope that it will be useful,
10  * but WITHOUT ANY WARRANTY; without even the implied warranty of
11  * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
12  * GNU General Public License for more details.
13  *
14  * You should have received a copy of the GNU General Public License
15  * along with this program; if not, write to the Free Software
16  * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
17  *
18  * Author: Andrea Lupia <alupia@dimes.unical.it>
19  */
20
21  #ifndef TRUST_TABLE_H
22  #define TRUST_TABLE_H
23
24  #include <map>
25  #include <set>
26  #include "ns3/simulator.h"
27  #include "ns3/object.h"
28  #include "ns3/ipv4-address.h"
29  #include "ns3/timer.h"
30  #include "ns3/callback.h"
31  #include "ns3/traced-value.h"
32
33  namespace ns3
```

```
{
namespace tms
{

enum Action
{
TRANSMISSION,
RECOMMENDATION // Not supported yet
};

struct TrustEntry
{
Ipv4Address m_agent;
Action m_action;
double m_value;
double m_numerator;
double m_denominator;
Time m_time;

TrustEntry (Ipv4Address agent) :
m_agent (agent), m_action (TRANSMISSION), m_value (0), m_numerator (1), m_denominator (2),
m_time (Simulator::Now ())
{
}

};

class TrustTable : public Object
{
public:
TrustTable ();
~TrustTable ();
static TypeId GetTypeId ();
TypeId GetInstanceTypeId () const;
void AddObservation (Ipv4Address agent, bool outcome, Time time = Simulator::Now(), Action action = TRANSMISSION);
double GetTrustValue (Ipv4Address agent);
bool IsTrustworthy (Ipv4Address agent);
void SetTrustUpdateInterval (Time t);
Time GetTrustUpdateInterval () const { return m_interval; }
void SetRho (double rho) { m_rho = rho; }
double GetRho () const { return m_rho; }
void SetEpsilon (double epsilon) { m_epsilon = epsilon; }
double GetEpsilon () const { return m_epsilon; }

void SetCallback (Callback<void, Ipv4Address> cb) { m_handleDistrust = cb; }
Callback<void, Ipv4Address> GetCallback () const { return m_handleDistrust; }

typedef void (* DistrustTracedCallback) (Ipv4Address);
typedef void (* TrustChangeTracedCallback) (Ipv4Address, double);
private:
void ComputeTrust (TrustEntry *entry);
void Refresh (TrustEntry *entry = 0);
double H (double p);
double Log2 (double x);

std::map<uint32_t, TrustEntry*> m_table;
std::set<uint32_t> m_distrusted;

Timer m_timer;

Callback<void, Ipv4Address> m_handleDistrust;

// Parameters
Time m_interval;
double m_rho;
double m_epsilon;

// Trace sources
TracedCallback<Ipv4Address> m_cDistrusted;
TracedCallback<Ipv4Address, double> m_trustChange;
};

```

```
} /* namespace tms */
} /* namespace ns3 */

#endif /* TRUST_TABLE_H */
```

trust-table.cc

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* Copyright (c) 2016 CULTURE TELELAB, DIMES - Universita' della Calabria
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
*
* Author: Andrea Lupia <alupia@dimes.unical.it>
*/

#include <math.h>
#include "trust-table.h"
#include "ns3/log.h"

namespace ns3
{
namespace tms
{

NS_LOG_COMPONENT_DEFINE ("TrustTable");

NS_OBJECT_ENSURE_REGISTERED (TrustTable);

TypeId
TrustTable::GetTypeId (void)
{
static TypeId tid = TypeId ("ns3::tms::TrustTable")
.SetParent<Object> ()
.SetGroupName ("Tms")
/*
* How to connect to this trace source:
*
* Config::Connect (
*  "/NodeList/0/$ns3::tms::TrustTable/Distrusted",
*  MakeCallback (&Class::Method, this)
* );
*
* void Class::Method (std::string context, Ipv4Address address)
* {
*   NS_LOG_UNCOND ("Traced " << context << " - " << Simulator::Now () << " - " << address);
* }
*/
.AddTraceSource ("Distrusted",
"Time and node distrusted",
MakeTraceSourceAccessor (&TrustTable::m_cDistrusted),
"ns3::tms::TrustTable::DistrustTracedCallback")
.AddTraceSource ("TrustChange",
"Time and node distrusted",
MakeTraceSourceAccessor (&TrustTable::m_trustChange),
"ns3::tms::TrustTable::TrustChangeTracedCallback")
;
```

```
62  return tid;
63  }
64
65  TypeId
66  TrustTable::GetInstanceTypeId (void) const
67  {
68  return GetTypeId ();
69  }
70
71  TrustTable::TrustTable () :
72  m_timer (Timer::CANCEL_ON_DESTROY), m_rho (0.666), m_epsilon (0.001)
73  {
74
75  }
76
77  TrustTable::~TrustTable ()
78  {
79
80  }
81
82  void
83  TrustTable::AddObservation (Ipv4Address agent, bool outcome, Time time, Action action)
84  {
85  NS_LOG_DEBUG ("Observation for agent " << agent << ": " << outcome);
86  std::map<uint32_t, TrustEntry*>::iterator i = m_table.find (agent.Get ());
87  TrustEntry *entry;
88  if (i == m_table.end ())
89  {
90  entry = new TrustEntry (agent);
91  m_table.insert (std::make_pair (agent.Get (), entry));
92  }
93  else
94  {
95  entry = i->second;
96  }
97  if (!m_timer.IsRunning ())
98  {
99  m_timer.Schedule ();
100 }
101 else if (Simulator::Now () > entry->m_time)
102 {
103 Refresh (entry);
104 }
105 entry->m_numerator += outcome;
106 entry->m_denominator++;
107 entry->m_time = Simulator::Now ();
108 ComputeTrust (entry);
109 }
110
111 void
112 TrustTable::ComputeTrust (TrustEntry *entry)
113 {
114 double p = entry->m_numerator / entry->m_denominator;
115 if (p <= 0)
116 entry->m_value = -1;
117 else if (p >= 1)
118 entry->m_value = 1;
119 else
120 entry->m_value = p >= 0.5 ? 1 - H (p) : H (p) - 1;
121 if (entry->m_value < - m_epsilon && m_distrusted.insert (entry->m_agent.Get ()).second)
122 {
123 NS_LOG_LOGIC ("Agent with address " << entry->m_agent << " has a negative trust value and becomes distrusted");
124 m_cDistrusted (entry->m_agent);
125 if (!m_handleDistrust.IsNull ())
126 m_handleDistrust (entry->m_agent);
127 }
128 else if (entry->m_value >= - m_epsilon && m_distrusted.erase (entry->m_agent.Get ()) == 1)
129 {
130 NS_LOG_LOGIC ("Agent with address " << entry->m_agent << " is now redeemed");
131 }
132 m_trustChange (entry->m_agent, entry->m_value);
133 NS_LOG_INFO ("New trust value for agent " << entry->m_agent << ": " << entry->m_value);
```

```
}

double
TrustTable::GetTrustValue (Ipv4Address agent)
{
std::map<uint32_t, TrustEntry*>::iterator i = m_table.find (agent.Get ());
if (i == m_table.end ())
return 0;
Refresh (i->second);
return i->second->m_value;
}

bool
TrustTable::IsTrustworthy (Ipv4Address agent)
{
return m_distrusted.find (agent.Get ()) == m_distrusted.end ();
// return GetTrustValue (agent) >= - m_epsilon;
}

void
TrustTable::SetTrustUpdateInterval (Time t)
{
m_interval = t;
m_timer.SetDelay (m_interval);
m_timer.SetFunction (&TrustTable::Refresh, this);
m_timer.SetArguments<TrustEntry*> (0);
}

void
TrustTable::Refresh (TrustEntry *entry)
{
if (entry == 0)
{
std::map<uint32_t, TrustEntry*>::iterator i = m_table.begin ();
for (; i != m_table.end (); i++)
Refresh (i->second);
m_timer.Schedule ();
}
else
{
NS_LOG_DEBUG ("Actualizing trust value of agent " << entry->m_agent);
entry->m_numerator--;
entry->m_denominator -= 2;
entry->m_numerator *= std::pow (m_rho, Simulator::Now ().GetSeconds () - entry->m_time.GetSeconds ());
entry->m_denominator *= std::pow (m_rho, Simulator::Now ().GetSeconds () - entry->m_time.GetSeconds ());
entry->m_numerator++;
entry->m_denominator += 2;
entry->m_time = Simulator::Now ();
ComputeTrust (entry);
}
}

double
TrustTable::H (double p)
{
return - p * Log2 (p) - (1 - p) * Log2 (1 - p);
}

double
TrustTable::Log2 (double x)
{
return std::log(x) / std::log (2.0);
}

} /* namespace tms */
} /* namespace ns3 */
```

tms-helper.h

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* Copyright (c) 2016 CULTURE TELELAB, DIMES - Universita' della Calabria
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
*
* Author: Andrea Lupia <alupia@dimes.unical.it>
*/

#ifndef TMS_HELPER_H
#define TMS_HELPER_H

#include "ns3/trust-manager.h"
#include "ns3/node-container.h"

namespace ns3 {

class TmsHelper {
public:
TmsHelper ();
~TmsHelper ();

/**
* \brief Enable TMS on a set of nodes
* \param nodes A NodeContainer holding the set of nodes to work with.
*/
void Install (NodeContainer nodes);
/**
* \brief Enable TMS on a single node
* \param node A Ptr<Node> to the node on which to enable TMS.
*/
void Install (Ptr<Node> node);
/**
* \brief Enable TMS on all nodes
*/
void InstallAll ();
};
}

#endif /* TMS_HELPER_H */
```

tms-helper.cc

```
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* Copyright (c) 2016 CULTURE TELELAB, DIMES - Universita' della Calabria
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
* published by the Free Software Foundation;
*
* This program is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
```

```
15 * along with this program; if not, write to the Free Software
16 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
17 *
18 * Author: Andrea Lupia <alupia@dimes.unical.it>
19 */
20
21 #include "tms-helper.h"
22 #include "ns3/pointer.h"
23 #include "ns3/node.h"
24 #include "ns3/node-list.h"
25 #include "ns3/ipv4-routing-protocol.h"
26 //#include "ns3/ipv4-l3-protocol.h"
27 //#include "ns3/ipv6-l3-protocol.h"
28
29 namespace ns3 {
30
31 TmsHelper::TmsHelper ()
32 {
33 }
34
35 TmsHelper::~TmsHelper ()
36 {
37 }
38
39 void
40 TmsHelper::Install (Ptr<Node> node)
41 {
42 Ptr<tms::TrustManager> trustManager = CreateObjectWithAttributes<tms::TrustManager> ("Node", PointerValue (node));
43 // FIXME Workaround to make stats with promiscuous mode
44 node->AggregateObject (trustManager);
45 Ptr<Ipv4RoutingProtocol> protocol = node->GetObject<Ipv4RoutingProtocol> ();
46 if (protocol)
47 {
48 protocol->AggregateObject (trustManager->GetTrustTable ());
49 }
50 else
51 {
52 NS_FATAL_ERROR ("Cannot find Ipv4RoutingProtocol installed on nodes");
53 }
54 // From the routing protocol:
55 // Ptr<tms::TrustManager> tms = GetObject<tms::TrustManager> ();
56
57 //Ptr<Ipv4L3Protocol> ipv4 = node->GetObject<Ipv4L3Protocol> ();
58 //if (ipv4)
59 //{
60 //Ptr<TrustManager> trustManager = Create<TrustManager> (node);
61 //}
62 //Ptr<Ipv6L3Protocol> ipv6 = node->GetObject<Ipv6L3Protocol> ();
63 //if (ipv6)
64 //{
65 // NOT SUPPORTED YET
66 //}
67 }
68
69 void
70 TmsHelper::Install (NodeContainer nodes)
71 {
72 for (NodeContainer::Iterator i = nodes.Begin (); i != nodes.End (); ++i)
73 {
74 Ptr<Node> node = *i;
75 //if (node->GetObject<Ipv4L3Protocol> () || node->GetObject<Ipv6L3Protocol> ())
76 //{
77 //Install (node);
78 //}
79 if (node->GetNDevices () > 0)
80 {
81 Install (node);
82 }
83 }
84 }
85
86 void
```

```
87  TmsHelper::InstallAll ()
88  {
89  for (NodeList::Iterator i = NodeList::Begin (); i != NodeList::End (); ++i)
90  {
91  Ptr<Node> node = *i;
92  //if (node->GetObject<Ipv4L3Protocol> () || node->GetObject<Ipv6L3Protocol> ())
93  //{
94  //Install (node);
95  //}
96  if (node->GetNDevices () > 0)
97  {
98  Install (node);
99  }
100 }
101 }
102
103 }
```

wscript

```
1   # -*- Mode: python; py-indent-offset: 4; indent-tabs-mode: nil; coding: utf-8; -*-
2
3   # def options(opt):
4   #     pass
5
6   # def configure(conf):
7   #     conf.check_nonfatal(header_name='stdint.h', define_name='HAVE_STDINT_H')
8
9   def build(bld):
10  module = bld.create_ns3_module('tms', ['wifi', 'internet'])
11  module.source = [
12  'model/trust-pmf.cc',
13  'model/trust-table.cc',
14  'model/trust-manager.cc',
15  'helper/tms-helper.cc',
16  ]
17
18  headers = bld(features='ns3header')
19  headers.module = 'tms'
20  headers.source = [
21  'model/trust-pmf.h',
22  'model/trust-table.h',
23  'model/trust-manager.h',
24  'helper/tms-helper.h',
25  ]
26
27  if bld.env.ENABLE_EXAMPLES:
28  bld.recurse('examples')
29
30  # bld.ns3_python_bindings()
```

A.2 Module *aodv* (diff)

The changes listed in this section have to be applied to the following files, all located under the "model" directory of the *aodv* module:

- *aodv-rtable.h*;
- *aodv-rtable.cc*;

- *aodv-routing-protocol.h*;
- *aodv-routing-protocol.cc*.

In the *wscript* file in the "aodv" main folder, the dependency to *tms* module needs to be added. The enhancements implemented with respect to the "vanilla" version of AODV protocol in NS-3.26 are the following:

- integration with *tms* module;
- capability for each node to drop a percentage of received data packets;
- HELLO packets broadcasting only when a node is part of an active route.

aodv-rtable.h

```
--- original\aodv-rtable.h
+++ modified\aodv-rtable.h
@@ -1,6 +1,7 @@
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* Copyright (c) 2009 IITP RAS
+ * Copyright (c) 2016 CULTURE TELELAB, DIMES - Universita' della Calabria
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -24,6 +25,8 @@
*
* Authors: Elena Buchatskaia <borovkovaes@iitp.ru>
*          Pavel Boyko <boyko@iitp.ru>
+ *
+ * Active routes enhancement by Andrea Lupia <alupia@dimes.unical.it>
*/
#ifndef AODV_RTABLE_H
#define AODV_RTABLE_H
@@ -31,10 +34,12 @@
#include <stdint.h>
#include <cassert>
#include <map>
+#include <set>
#include <sys/types.h>
#include "ns3/ipv4.h"
#include "ns3/ipv4-route.h"
#include "ns3/timer.h"
+#include "ns3/callback.h"
#include "ns3/net-device.h"
#include "ns3/output-stream-wrapper.h"

@@ -229,6 +234,12 @@
*  3. The Lifetime field is updated to current time plus DELETE_PERIOD.
*/
void InvalidateRoutesWithDst (std::map<Ipv4Address, uint32_t> const & unreachable);
+  /// Update active routes (used for hello transmission)
+  void UpdateActiveRoutes (RoutingTableEntry & rt);
+  /// Set callback for in active route
+  void SetActiveRouteCallback (Callback<void> cb) { m_arc = cb; }
+  /// Set callback for not in active route
+  void SetNoActiveRouteCallback (Callback<void> cb) { m_narc = cb; }
/// Delete all route from interface with address iface
void DeleteAllRoutesFromInterface (Ipv4InterfaceAddress iface);
/// Delete all entries from routing table
```

```
@@ -248,8 +259,12 @@
std::map<Ipv4Address, RoutingTableEntry> m_ipv4AddressEntry;
/// Deletion time for invalid routes
Time m_badLinkLifetime;
-  /// const version of Purge, for use by Print() method
-  void Purge (std::map<Ipv4Address, RoutingTableEntry> &table) const;
+  /// Set of active routes (used for hello transmission)
+  std::set<Ipv4Address> m_activeRoutes;
+  /// Callback for in active route
+  Callback<void> m_arc;
+  /// Callback for not in active route
+  Callback<void> m_narc;
};

}
```

aodv-rtable.cc

```
--- original\aodv-rtable.cc
+++ modified\aodv-rtable.cc
@@ -1,6 +1,7 @@
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* Copyright (c) 2009 IITP RAS
+ * Copyright (c) 2016 CULTURE TELELAB, DIMES - Universita' della Calabria
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -24,6 +25,8 @@
*
* Authors: Elena Buchatskaia <borovkovaes@iitp.ru>
*          Pavel Boyko <boyko@iitp.ru>
+ *
+ * Active routes enhancement by Andrea Lupia <alupia@dimes.unical.it>
*/

#include "aodv-rtable.h"
@@ -236,6 +239,9 @@
{
NS_LOG_FUNCTION (this << dst);
Purge ();
+  // DeleteRoute called only when no route is found after a route request, deleted entry is never in VALID state
+  RoutingTableEntry rt;
+  NS_ASSERT (LookupValidRoute (dst, rt) == false);
if (m_ipv4AddressEntry.erase (dst) != 0)
{
NS_LOG_LOGIC ("Route deletion to " << dst << " successful");
@@ -254,6 +260,7 @@
rt.SetRreqCnt (0);
std::pair<std::map<Ipv4Address, RoutingTableEntry>::iterator, bool> result =
m_ipv4AddressEntry.insert (std::make_pair (rt.GetDestination (), rt));
+  UpdateActiveRoutes (rt);
return result.second;
}

@@ -268,6 +275,7 @@
NS_LOG_LOGIC ("Route update to " << rt.GetDestination () << " fails; not found");
return false;
}
+  UpdateActiveRoutes (rt);
i->second = rt;
if (i->second.GetFlag () != IN_SEARCH)
{
@@ -290,6 +298,7 @@
}
i->second.SetFlag (state);
i->second.SetRreqCnt (0);
+  UpdateActiveRoutes (i->second);
```

```
NS_LOG_LOGIC ("Route set entry state to " << id << ": new state is " << state);
return true;
}
@@ -326,8 +335,29 @@
{
NS_LOG_LOGIC ("Invalidate route with destination address " << i->first);
i->second.Invalidate (m_badLinkLifetime);
+              UpdateActiveRoutes (i->second);
}
}
+    }
+}
+
+void
+RoutingTable::UpdateActiveRoutes (RoutingTableEntry & rt)
+{
+  NS_LOG_FUNCTION (this);
+  if ((rt.GetFlag () != VALID || rt.IsPrecursorListEmpty () == true)
+      && m_activeRoutes.erase (rt.GetDestination ()) > 0
+      && m_activeRoutes.empty () == true && m_narc.IsNull () == false)
+    {
+      NS_LOG_LOGIC ("Calling not in active route callback");
+      m_narc ();
+    }
+  else if (rt.GetFlag () == VALID && rt.IsPrecursorListEmpty() == false
+      && m_activeRoutes.insert (rt.GetDestination ()).second == true
+      && m_activeRoutes.size () == 1 && m_arc.IsNull () == false)
+    {
+      NS_LOG_LOGIC ("Calling in active route callback");
+      m_arc ();
}
}

@@ -372,39 +402,7 @@
{
NS_LOG_LOGIC ("Invalidate route with destination address " << i->first);
i->second.Invalidate (m_badLinkLifetime);
-              ++i;
-            }
-          else
-            ++i;
-        }
-      else
-        {
-          ++i;
-        }
-    }
-}
-
-void
-RoutingTable::Purge (std::map<Ipv4Address, RoutingTableEntry> &table) const
-{
-  NS_LOG_FUNCTION (this);
-  if (table.empty ())
-    return;
-  for (std::map<Ipv4Address, RoutingTableEntry>::iterator i =
-         table.begin (); i != table.end ();)
-    {
-      if (i->second.GetLifeTime () < Seconds (0))
-        {
-          if (i->second.GetFlag () == INVALID)
-            {
-              std::map<Ipv4Address, RoutingTableEntry>::iterator tmp = i;
-              ++i;
-              table.erase (tmp);
-            }
-          else if (i->second.GetFlag () == VALID)
-            {
-              NS_LOG_LOGIC ("Invalidate route with destination address " << i->first);
-              i->second.Invalidate (m_badLinkLifetime);
+              UpdateActiveRoutes (i->second);
++i;
```

```
}
else
@@ -439,7 +437,6 @@
RoutingTable::Print (Ptr<OutputStreamWrapper> stream) const
{
std::map<Ipv4Address, RoutingTableEntry> table = m_ipv4AddressEntry;
-  Purge (table);
*stream->GetStream () << "\nAODV Routing table\n"
<< "Destination\tGateway\t\tInterface\tFlag\tExpire\t\tHops\n";
for (std::map<Ipv4Address, RoutingTableEntry>::const_iterator i =
```

aodv-routing-protocol.h

```
--- original\aodv-routing-protocol.h
+++ modified\aodv-routing-protocol.h
@@ -1,6 +1,7 @@
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* Copyright (c) 2009 IITP RAS
+ * Copyright (c) 2016 CULTURE TELELAB, DIMES - Universita' della Calabria
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -24,6 +25,8 @@
*
* Authors: Elena Buchatskaia <borovkovaes@iitp.ru>
*          Pavel Boyko <boyko@iitp.ru>
+ *
+ * Expanding ring search and TTL fix: Andrea Lupia <alupia@dimes.unical.it>
*/
#ifndef AODVROUTINGPROTOCOL_H
#define AODVROUTINGPROTOCOL_H
@@ -86,6 +89,8 @@
bool GetHelloEnable () const { return m_enableHello; }
void SetBroadcastEnable (bool f) { m_enableBroadcast = f; }
bool GetBroadcastEnable () const { return m_enableBroadcast; }
+  void SetTrustEnable (bool f) { m_enableTrust = f; }
+  bool GetTrustEnable () const { return m_enableTrust; }

/**
* Assign a fixed random variable stream number to the random variables
@@ -138,6 +143,12 @@
bool m_gratuitousReply;              ///< Indicates whether a gratuitous RREP should be sent
bool m_enableHello;                  ///< Indicates whether a hello messages enable
bool m_enableBroadcast;              ///< Indicates whether a a broadcast data packets forwarding enable
+
+  // Malicious node attributes
+  double m_dropPercentage;
+
+  // Trust management attribute
+  bool m_enableTrust;
//\}

/// IP protocol
@@ -269,6 +280,10 @@
void RouteRequestTimerExpire (Ipv4Address dst);
/// Mark link to neighbor node as unidirectional for blacklistTimeout
void AckTimerExpire (Ipv4Address neighbor,  Time blacklistTimeout);
+  /// Schedule hello transmission
+  void ScheduleHello ();
+  /// Cancel hello transmission
+  void CancelHello ();

/// Provides uniform random variables.
Ptr<UniformRandomVariable> m_uniformRandomVariable;
```

aodv-routing-protocol.cc

```
--- original\aodv-routing-protocol.cc
+++ modified\aodv-routing-protocol.cc
@@ -1,6 +1,7 @@
/* -*- Mode:C++; c-file-style:"gnu"; indent-tabs-mode:nil; -*- */
/*
* Copyright (c) 2009 IITP RAS
+ * Copyright (c) 2016 CULTURE TELELAB, DIMES - Universita' della Calabria
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -24,6 +25,8 @@
*
* Authors: Elena Buchatskaia <borovkovaes@iitp.ru>
*          Pavel Boyko <boyko@iitp.ru>
+ *
+ * Expanding ring search and TTL fix: Andrea Lupia <alupia@dimes.unical.it>
*/
#define NS_LOG_APPEND_CONTEXT                                   \
if (m_ipv4) { std::clog << "[node " << m_ipv4->GetObject<Node> ()->GetId () << "] "; }
@@ -41,6 +44,7 @@
#include "ns3/adhoc-wifi-mac.h"
#include "ns3/string.h"
#include "ns3/pointer.h"
+#include "ns3/trust-table.h"
#include <algorithm>
#include <limits>

@@ -143,6 +147,8 @@
m_destinationOnly (false),
m_gratuitousReply (true),
m_enableHello (false),
+  m_dropPercentage (0.),
+  m_enableTrust (false),
m_routingTable (m_deletePeriod),
m_queue (m_maxQueueLen, m_maxQueueTime),
m_requestId (0),
@@ -277,6 +283,17 @@
StringValue ("ns3::UniformRandomVariable"),
MakePointerAccessor (&RoutingProtocol::m_uniformRandomVariable),
MakePointerChecker<UniformRandomVariable> ())
+    .AddAttribute ("DropPercentage",
+                   "Drop percentage, if >0 node is malicious",
+                   DoubleValue (0.),
+                   MakeDoubleAccessor (&RoutingProtocol::m_dropPercentage),
+                   MakeDoubleChecker<double> ())
+    .AddAttribute ("EnableTrust",
+                   "Enable trust management",
+                   BooleanValue (false),
+                   MakeBooleanAccessor (&RoutingProtocol::SetTrustEnable,
+                                        &RoutingProtocol::GetTrustEnable),
+                   MakeBooleanChecker ())
;
return tid;
}
@@ -389,6 +406,10 @@
}
UpdateRouteLifeTime (dst, m_activeRouteTimeout);
UpdateRouteLifeTime (route->GetGateway (), m_activeRouteTimeout);
+      if (dst.IsBroadcast () == false && dst != rt.GetInterface ().GetBroadcast ())
+        {
+          m_nb.Update (route->GetGateway (), m_activeRouteTimeout);
+        }
return route;
}

@@ -561,6 +582,21 @@
return true;
}
```

```
+  // Malicious drop code
+  Ptr<Packet> pkt = p->Copy ();
+  UdpHeader udpHeader;
+  pkt->PeekHeader (udpHeader);
+  if (udpHeader.IsChecksumOk ())
+    {
+      pkt->RemoveHeader (udpHeader);
+    }
+  TypeHeader aodvHeader;
+  pkt->PeekHeader(aodvHeader);
+  if (!aodvHeader.IsValid () && m_uniformRandomVariable->GetValue (0., 1.) < m_dropPercentage)
+    {
+      return true;
+    }
+
// Forwarding
return Forwarding (p, header, ucb, ecb);
}
@@ -1184,6 +1220,16 @@
}
}

+  if (m_enableTrust)
+    {
+      // TMS-> If node is not trustworthy, RREQ is not taken into account
+      if (GetObject<ns3::tms::TrustTable> ()->IsTrustworthy (src) == false)
+        {
+          NS_LOG_DEBUG ("Node is not trustworthy, ignoring RREQ");
+          return;
+        }
+    }
+
uint32_t id = rreqHeader.GetId ();
Ipv4Address origin = rreqHeader.GetOrigin ();

@@ -2052,15 +2098,42 @@
RoutingProtocol::DoInitialize (void)
{
NS_LOG_FUNCTION (this);
+  if (m_enableHello)
+    {
+      m_htimer.SetFunction (&RoutingProtocol::HelloTimerExpire, this);
+      m_routingTable.SetActiveRouteCallback (MakeCallback (&RoutingProtocol::ScheduleHello, this));
+      m_routingTable.SetNoActiveRouteCallback (MakeCallback (&RoutingProtocol::CancelHello, this));
+    }
+  if (m_enableTrust)
+    {
+      Ptr<ns3::tms::TrustTable> trust = GetObject<ns3::tms::TrustTable> ();
+      if (trust)
+        {
+          trust->SetCallback (MakeCallback (&RoutingProtocol::SendRerrWhenBreaksLinkToNextHop, this));
+        }
+      else
+        {
+          NS_FATAL_ERROR ("Cannot enable trust (trust table is missing)");
+        }
+    }
+  Ipv4RoutingProtocol::DoInitialize ();
+}
+
+void
+RoutingProtocol::ScheduleHello ()
+{
+  NS_LOG_FUNCTION (this);
uint32_t startTime;
-  if (m_enableHello)
-    {
-      m_htimer.SetFunction (&RoutingProtocol::HelloTimerExpire, this);
-      startTime = m_uniformRandomVariable->GetInteger (0, 100);
-      NS_LOG_DEBUG ("Starting at time " << startTime << "ms");
-      m_htimer.Schedule (MilliSeconds (startTime));
-    }
```

```
-  Ipv4RoutingProtocol::DoInitialize ();
+  startTime = m_uniformRandomVariable->GetInteger (0, 100);
+  NS_LOG_DEBUG ("Starting in " << startTime << "ms");
+  m_htimer.Schedule (MilliSeconds (startTime));
+}
+
+void
+RoutingProtocol::CancelHello ()
+{
+  NS_LOG_FUNCTION (this);
+  m_htimer.Cancel ();
}

} //namespace aodv
```