FOURTH QUARTER 2005, VOLUME 7, NO. 4

IEEE COMMUNICATIONS SURVEYS
The Electronic Magazine of Original Peer-Reviewed Survey Articles

www.comsoc.org/pubs/surveys

A SURVEY OF SECURITY ISSUES IN MOBILE AD HOC AND SENSOR NETWORKS

DJAMEL DJENOURI AND LYES KHELLADI, CERIST CENTER OF RESEARCH, ALGIERS
NADJIB BADACHE, UNIVERSITY OF SCIENCE AND TECHNOLOGY, ALGIERS

ABSTRACT

Security in mobile ad hoc networks is difficult to achieve, notably because of the vulnerability of wireless links, the limited physical protection of nodes, the dynamically changing topology, the absence of a certification authority, and the lack of a centralized monitoring or management point. Earlier studies on mobile ad hoc networks (MANETs) aimed at proposing protocols for some fundamental problems, such as routing, and tried to cope with the challenges imposed by the new environment. These protocols, however, fully trust all nodes and do not consider the security aspect. They are consequently vulnerable to attacks and misbehavior.

More recent studies focused on security problems in MANETs, and proposed mechanisms to secure protocols and applications. This article surveys these studies. It presents and discusses several security problems along with the currently proposed solutions (as of July 2005) at different network layers of MANETs. Security issues involved in this article include routing and data forwarding, medium access, key management and intrusion detection systems (IDSs). This survey also includes an overview of security in a particular type of MANET, namely, wireless sensor networks (WSNs).

Nowadays, with the rapid proliferation of wireless lightweight devices such as laptops, PDAs, wireless telephones, and wireless sensors, the potential and importance of nomadic computing, and particularly mobile ad hoc networking, have become apparent.

Some applications of mobile networks could not support the dependence on any fixed infrastructure. As examples of such applications we cite: emergency disaster relief in a damaged area after a storm or an earthquake; a set of digital sensors positioned to take measurements in a region unreachable by humans; military tanks and planes in a battlefield; and finally, students (or researchers) sharing information during a lecture (or conference). This infrastructure independency requirement leads to a new kind of mobile network, namely, *ad hoc* networks.

A mobile ad hoc network, or MANET, is a temporary infrastructureless network, formed by a set of wireless mobile hosts that dynamically establish their own network *on the fly*, without relying on any central administration.

Mobile hosts used in MANETs must ensure the roles that are ensured by the powerful fixed infrastructure in traditional networks. This is a challenging task, since these devices have limited resources (CPU, storage, energy, etc.). Moreover, the network's environment has some features that add extra complications, such as the frequent topology changes caused by nodes' mobility, as well as the unreliability and the bandwidth limitation of wireless channels.

Earlier studies on ad hoc networks aimed to propose solutions for some fundamental problems, coping with the new challenges caused by the features cited above. Nevertheless, these solutions should be secure and tamper-resistant in order to ensure the proper functioning of the system, and to provide a tolerable quality of service in such an open vulnerable environment. More recent studies have focused on security problems in MANETs, proposing mechanisms and techniques to protect the basic protocols and applications.

Our contribution in this article is to survey *several* security issues in MANETs by covering *different network layers*. The rest of the article is organized as follows. First we present some basic concepts, followed by security issues regarding routing protocols and those regarding data forwarding at the same layer (network). We will also deal with MAC-layer security issues. Afterwards, the key management will be presented (which is a basic framework essential to secure application as well as underlying protocols), followed by MANETs' intrusion detection systems (IDSs). We devote a section to a newly

emergent type of ad hoc networking, i.e. sensor networks, where different security issues related to this particular application will be presented. Finally, we conclude the article.

Basic Concepts

Security Requirements

The security services of ad hoc networks are not altogether different from those of other network. The goal of these services is to protect information and resources from attacks and misbehavior. In dealing with network security, we will explain the following requirements that an effective security architecture must ensure:

Availability: Ensures that the desired network services are available whenever they are expected, in spite of the presence of attacks. Systems that ensure availability in MANETs seek to combat denial of service and energy starvation attacks, as well as node misbehavior such as node selfishness in packet forwarding. All these threats will be presented later.

Authentication: Ensures that communication from one node to another is genuine. In other words, it ensures that a malicious node cannot masquerade as a trusted network node.

Data confidentiality: Ensures that a given message cannot be understood by anyone other than its (their) desired recipient(s). Data confidentiality is typically enabled by applying symmetric or asymmetric data encryption.

Integrity: Denotes the *authenticity of data* sent from one node to another. That is, it ensures that a message sent from node A to node B was not modified by any malicious node C during its transmission. If a robust confidentiality mechanism is employed, ensuring data integrity may be as simple as adding one-way hashes [1] before encrypting messages.

Non-repudiation: In computer networks, non-repudiation is the ability to ensure that a node cannot deny the sending of a message that it originated. Digital signatures [1] may be used to ensure this.

MANET Features and Their Impact on Security

The following features make MANETs more vulnerable than traditional networks.

Infrastructureless: Central servers, specialized hardware, and fixed routers are necessarily absent. The lack of such infrastructure precludes the deployment of centralized host relationships. Instead, nodes uphold egalitarian relationships, that is, any security solution should rely on a distributed cooperative scheme instead of a centralized scheme.

Wireless link use: Wireless link usage renders ad hoc networks susceptible to attacks. Unlike wired networks, in which an adversary must gain physical access to the network's wires or pass through several lines of defense at firewalls and gateways, attacks on a wireless ad hoc network can come from all directions and target any node. Hence, ad hoc networks will not have a clear line of defense, and every node must be prepared to defend against threats. Moreover, the MAC protocols used in ad hoc networks, such IEEE802. 11, rely on *trusted* cooperation in a neighborhood to ensure channel access, which leads to high vulnerability.

Multi-hop: Because of the lack of central routers and gateways, hosts are themselves routers. Thus, packets follow *multi-hop* routes and pass through different mobile nodes before arriving at their final destination. Due to the possible untrustworthiness of such nodes, this feature presents a serious vulnerability.

Node movement autonomy: Mobile nodes are generally autonomous units that are capable of *roaming* independently. This means that tracking down a particular mobile node in a large-scale ad hoc network cannot be done easily.

Amorphous: Node mobility and wireless connectivity allow nodes to enter and leave the network spontaneously, to form and break links unintentionally. Therefore, the network topology has no *fixed* form regarding both its size and shape, i.e., it changes frequently. Any security solution must take this feature into account.

Power limitation: Ad hoc enabled mobile hosts are small and lightweight, and they are often supplied with limited power resources, such as small batteries. This limitation causes a vulnerability, namely, attackers may target some nodes' batteries to disconnect them, which may lead to a network partition. This is called an energy starvation attack or sleep deprivation torture attack [2]. This feature also represents a challenging constraint when designing security solutions for MANETs.

Memory and computation power limitation: Ad hoc enabled mobile nodes have limited storage devices and weak computational capabilities. Consequently, high complexity security solutions, such as symmetric or asymmetric data encryption, are difficult to implement.

Mobile devices physical vulnerability: Mobile devices used in MANETs, and in mobile networks in general, are lightweight and portable. This represents a vulnerability, since the devices and the information stored in the devices can be easily stolen. Mechanisms for protecting both devices and information should be employed.

Threats

We divide threats that can affect security in ad hoc networks into two classes, attacks and misbehavior.

Attacks

Attacks include any action that *intentionally* aims to cause any damage to the network. They can be divided according to their origin or their nature. An origin-based classification splits attacks into two categories, external and internal, whereas a nature-based classification splits them into passive attacks and active attacks.

External attacks: Includes attacks launched by a node that does not belong to the *logical* network, or is not allowed to access to it.

Internal attacks: Includes attacks launched by an internal compromised or malicious node. This is a more severe type of threat since the proposed defense toward external attacks is ineffective against compromised and internal malicious nodes.

Passive attacks: A passive attack is a continuous collection of information that might be used later when launching an active attack. For that, the attacker eavesdrops packets and analyzes them to pick up required information. Due to the nature of the wireless communication medium which is widely shared, it is easier for an attacker to launch such an attack in this environment than in traditional wired environments. The security attribute that must be provided here is *information confidentiality*.

Active attacks: Includes almost all other attacks launched by actively interacting with victims, such as: *sleep deprivation torture,* which targets the batteries; *hijacking*, in which the attacker takes control of a communication between two entities and masquerades as one of them; *jamming*, which causes channel unavailability by overusing it, attacks against routing protocols that we will see in the next section, etc. Most of these attacks result in a denial of service (DoS), which is a degradation or a complete halt in communication between nodes.

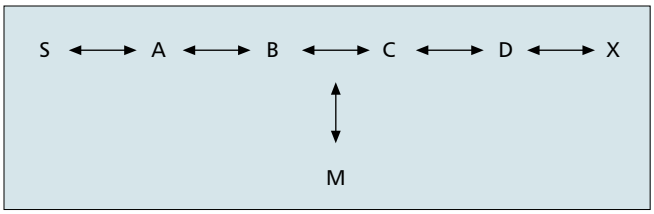

Figure 1. *Example of an ad hoc network.*

Misbehavior

We define misbehavior threats as an unauthorized behavior of an *internal* node that can result *unintentionally* in damage to other nodes, i.e., the aim of the node is not to launch an attack, but it may have other aims such as obtaining an unfair advantage compared with the other nodes. For instance, one may do not correctly execute the MAC protocol, with the intent of getting higher bandwidth, or it may refuse to forward packets for others to save its resources, while using their resources and asking them to forward its own packets.

Up to now we have presented basic concepts regarding security in MANETs. In the following sections we will deal with the current research areas related to security in MANETs, and we will discus existing problems and proposed solutions.

Routing Security Issues

A MANET's routing protocol finds routes between nodes over which data packets are forwarded toward the final destination. In contrast to traditional network routing protocols, MANET routing protocols must be adaptable to cope with the features presented previously, especially the frequent changes in network topology. The challenging problem of routing in ad hoc networks has been extensively studied, particularly in the MANET working group of the Internet Engineering Task Force (IETF) [3]. These studies have resulted in several mature protocols [4–10], which can be divided into two classes: proactive (table driven) and reactive (on-demand). (A survey of the two classes of routing protocols is available in [11].) It has been shown in [12] that reactive protocols are more adaptable to MANET environments than proactive protocols. However, the problem with all of these solutions is that they trust all nodes and do not account for security, therefore they are vulnerable to attacks.

It is highly important to secure the routing protocol. If the routing protocol can be subverted and messages can be altered in transit, then no amount of security on the data packets at the upper layers can mitigate threats. Recently, several secure MANET routing protocols have been proposed [13–23]. Some of these solutions have been surveyed in [24]. In this section we deal with the security issues of routing protocols. After an overview of DSR and AODV, two routing protocols involved in this section, we will present a classification of different attacks that threat traditional MANET routing protocols, and we will discuss recent proposed solutions.

DSR and AODV in a Nutshell

In the following sections we give general descriptions of DSR and AODV, two protocols largely adopted by IETF's MANET working group [3]. An overview of these two protocols is essential, since the attacks presented later are analyzed in terms of these protocols.

DSR (Dynamic Source Routing)

DSR [4] is a reactive protocol based on the source route approach. The principal of this approach is that the whole route is chosen by the source, and is put within each packet sent. Each node keeps in its cache the source routes learned. When it needs to send a packet, it first checks in its cache for the existence of such a route. If no entry to the appropriate destination is available in the cache, then the node launches a route discovery by broadcasting a request (RREQ) packet through the network. When receiving the (RREQ), a node seeks a route in its cache for the RREQ's destination; finding such a route results in sending a route reply (RREP) packet to the source. However, if no appropriate route exists then the node adds its address to the RREQ and continues broadcasting. When a node detects a route failure, it sends a route error (RER) packet to the source that uses this link, then this one applies again the route discovery process.

AODV (Ad hoc On-Demand Distance Vector)

AODV [7] is a hop-by-hop routing protocol. When a node needs to send a data packet to a destination to which it has no route, it has to broadcast a RREQ to all its neighbors, then each neighbor does so until reaching the destination (or a node with a valid route to the destination). This node sends a RREP packet that travels the *inverse* path until reaching the source. Upon the reception of this reply each intermediary updates its routing table. In this way a route between the source and the destination is built. Unlike in DSR, the source does not put the whole route within the outgoing packets; rather, the decision about the next hop is made separately after each hop. Since it relies on the distance vector principle [7], AODV assigns monotonically increasing sequence numbers to routes, which defines route freshness, as well as hop-count, which defines route optimality.

Routing Protocol Attacks

The earlier proposed routing protocols for MANETs are subject to many different types of attacks. Analogous exploits exist in wired networks [25], but can be more easily overcome by the existing powerful infrastructure. In this subsection we present and analyze several classes of attacks against ad hoc routing protocols. Without loss of generality, these attacks are discussed in terms of AODV and DSR protocols, which are used as representatives of ad hoc on-demand protocols. However, almost all traditional on-demand protocols have the same vulnerabilities. We think the table-driven approach is unsuitable for MANETs, so it is excluded from the study.

Attacks using Modification

Network traffic can be redirected and DoS attacks can be launched by *modifying* routing information [19], such as *altering* control message fields of data packets or forwarding routing messages with *falsified* values. We now detail several attacks that use modification.

Redirection by modifying route sequence numbers: Some routing protocols, such as AODV [7], instantiate and maintain routes by assigning monotonically increasing sequence numbers to routes. Consequently, any node may divert traffic through itself by claiming a route with a sequence number higher than the authentic value. Consider the example illustrated in Fig. 1 [19]. Suppose that a malicious node M receives the RREQ originated from S for destination X after it is rebroadcast by B during a route discovery. M can redirect traffic toward itself by unicasting to B a RREP containing a much higher sequence number for X than the value last advertised by X. Eventually, the RREQ broadcast by B will reach a node with a valid route to X and a valid RREP will be unicast back toward S. However, at that point B will have already received the false RREP from M. If the sequence number for X that

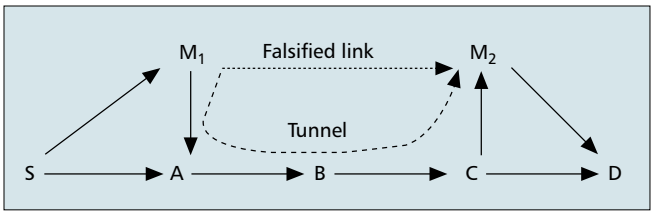

Figure 2. *Tunneling.*

M used in the false RREP is higher than the sequence number for X in the valid RREP, B will drop the valid RREP, thinking that the valid route is stale. Consequently, all subsequent traffic destined for X that travels through B will be directed toward M. The situation will not be corrected until either a legitimate RREQ or a legitimate RREP for X with a higher sequence number enters the network.

Redirection by modifying hop-count: Many traditional ad hoc routing protocols, such as AODV, use the hop-count field to determine the optimal path. Consequently, malicious nodes can increase their chances to be included on a newly created route by resetting the hop count field of the RREQ they forward to zero. Such an attack is most threatening when combined with spoofing, as detailed later. The redirection attack is possible even if the protocol uses metrics other than the hops number. In this case all the attacker has to do is modify the field used to compute the metric instead of the hop-count.

Source routes modification: As we have previously seen, DSR [4] utilizes the source routing strategy, thereby source nodes explicitly state routes in data packets. These routes lack any integrity checks, so alterations of source routes in packet headers can be easily performed by malicious nodes, resulting in denial-of-service attacks. Assume a path exists from S to X as shown in Fig. 1. Also assume that C and X are out of the power range of each other, and that M is a malicious node attempting a denial-of-service attack. Suppose S wishes to communicate with X to which it has an unexpired route in its route cache. S transmits a data packet toward X with the source route (S, A, B, M, C, D, X) attached to the packet's header. When M receives the packet, it can alter the source route in the packet's header, e.g. it can delete D from the source route. Consequently, when C receives the altered packet, it attempts to forward it to X. Since X is out of C's power range, the packet will not reach X.

C will then consider that the link with X has been broken, and will send a RER packet back to S via M. When M receives this packet, it will simply drop it. Therefore, S will still use the route through M, and this latter will continue performing this way, resulting in a denial-of-service attack against the routing service. This attack can also be used to cause sleep deprivation (paragraph 2.3.1), since packets will be transmitted and *retransmitted* through compromised routes.

Tunneling: Two remote nodes may collaborate to encapsulate and exchange messages between them through existing data routes, and make impressions as they are adjacent. Therefore, they may collaborate to falsely represent the length of available paths by encapsulating and tunneling between them legitimate routing messages generated by other nodes, preventing the intermediate nodes from correctly incrementing the metric used to measure path lengths (such as hop count). For example, in Fig. 2 M_1 and M_2 are malicious nodes that use the path (M_1, A, B, C, M_2) as a tunnel. When M_1 receives a route request packet (RREQ) from S, it encapsulates and tunnels it to M_2, which the latter normally forwards. After M_2 gets the RREP from D, it tunnels it back to M_1, which does so to S, resulting in the construction of a short wrong route (S, M_1, M_2, D) that may be considered as the most optimal one.

Spoofing Attacks

Spoofing occurs when a node *misrepresents its identity* in the network, such as by altering its MAC or IP address in the outgoing packets. This attack can be readily combined with modification attacks. These two attacks (spoofing and modification), when combined with each other, may result in serious misinformation, such as creating route loops [19].

Attacks using Fabrication

This class includes attacks based on the generation of false routing messages. Such attacks are difficult to detect.

Falsifying route errors: On-demand routing protocols, such as AODV and DSR, implement path maintenance to recover broken paths caused by node mobility. As illustrated before, when a link of an active route from node S to node D breaks down, the upstream node (predecessor) of the link sends S back a RER packet. If this latter has no other route to D and a route is still needed to this destination, it initiates a new route discovery. The vulnerability is that routing attacks can be launched by disseminating false RER, resulting in the destruction of valid routes and extra overhead, which can cause DoS and sleep deprivation.

Broadcast falsified routes: In DSR a node can update its routing table (route cache) by relying on information in headers held by packets it forwards. Routes can also be learned from promiscuously received packets. The vulnerability is that an attacker could easily exploit this method of learning routes and then poison the route caches of its neighbors by broadcasting packets containing falsified routes.

Rushing Attacks

Recently, Hu *et al.* [21] have defined a new attack called a rushing attack. In almost all on-demand routing protocols, to limit the route discovery overhead each node forwards only one RREQ originated from any route discovery, generally the first one received. This property can be exploited by *rushing* the forwarding of received RREQs.

For a route discovery, if the RREQs forwarded by the attacker are the first to reach each neighbor of the target, then any route obtained by this route discovery will include the attacker. That is, when a neighbor of the target receives the rushed RREQ from the attacker, it forwards that RREQ, and will not forward any further RREQ from this route discovery. As a result, the initiator will be unable to discover any usable routes (i.e., routes that do not include the attacker) containing at least two hops (three nodes). In general terms, an attacker that can forward RREQs more quickly than legitimate nodes can launch such an attack and include itself in all the discovered routes. The rushing attack can also be used against any protocol that predictably forwards any *particular* RREQ for each route discovery. The attacker does the same thing, except that packets it sends must fit the appropriate feature [21]. However, in the following discussion we assume that nodes forward the first received RREQ.

How can an attacker rush RREQ packets to launch a rushing attack? A malicious node can use one or more of the following techniques:

- Remove MAC and/or network delays when forwarding packets: MAC-layer and network-layer protocols use delays on packet transmission to avoid collisions. Thus, an attacker may deny these delays to rush the request it forwards.
- Transmit RREQs in higher power: An attacker supplied with a powerful physical communication support can use a higher transmission power to forward RREQs, thereby covering a higher power range than other nodes, and further nodes will be reached in less hops. Unlike the other

Attack	AODV	DSR
Attacks using modification		
Modifying route sequence numbers	Yes	No
Modifying hop counts	Yes	No
Modifying source route	No	Yes
Tunneling	Yes	Yes
Spoofing attacks	Yes	Yes
Attacks using fabrication		
Falsifying route errors	yes	Yes
Broadcast falsified routes	No	Yes
Rushing attacks	Yes	Yes

Table 1. *Vulnerabilities of AODV and DSR.*

techniques, this technique does not, in general, allow the attacker to include itself in a unique discovered route, since it could not receive the RREP (except in the case where it has a highly sensitive receiver). However, it precludes discovery of valid routes.

- Employing the wormhole technique [26]: Two attackers can use a high-quality tunnel to pass a RREQ packet among them, allowing it to reach its final destination before the other RREQs. This can be done when one node is closer to the source and the other is closer to the destination, and a path with high quality exists between the two nodes (e.g., via a wired network). Note that a special high-quality route is required for this attack, which is different from the tunneling attack presented before that uses wireless multi-hop routes.

Table 1 provides a summary of DSR and AODV vulnerabilities to the attacks presented in this section. As shown, AODV is vulnerable to sequence number and hop-count modification attacks, since it is based on distance vector. On the other hand, DSR, which is based on source routing and uses promiscuous route learning, is sensitive to source route modification and to falsified routes broadcast. Both protocols are vulnerable to tunneling, spoofing, falsified route error fabrication, and rushing attacks.

SOLUTIONS

Authentication During All Routing Phases

This solution consists of using authentication techniques during all routing phases, to exclude attackers and unauthorized nodes from participating in the routing. Most of the proposed solutions belonging to this class modify existing routing protocols to build authentication-based solutions [13, 16, 19, 22, 27]. Since they use digital signatures, these solutions rely on a certificate authority (CA), such as the solution presented in [19], which requires the use of a trusted certificate server whose public key is *a priori* known to all valid nodes. This reliance on a fixed server renders the solution centralized and less flexible. The major advantage of this approach is that it excludes external unauthorized nodes from participating in the routing, therefore all the attacks presented previously are prevented when launched by an external node. Moreover, some of the attacks launched by an authorized node can be beaten, as illustrated in Table 2.

Trust-Level Metric

Yi *et al.* [22] define a new metric called trust value that governs routing protocol behavior. This metric is to be embedded into control packets to mirror the minimum trust value required by the sender. Thus, a node that receives any packet can neither process it nor forward it unless it provides the required trust level presented in the packet. In this manner, the authors designed SAR (Security-Aware Routing), a protocol derived from AODV and based on the hierarchal trust values metric and authentication. In SAR, this metric is also used as a criterion to select routes when many routes satisfying the required trust value are available. To define nodes' trust values, the authors address the example of a military context, in which the trust level matches to node's owner rank. However, in the general context, where there is no hierarchy in the network, defining the nodes' trust values is problematic. This technique is based on authentication and requires a hier-

Solutions	Attacks prevented	Drawbacks
Authentication during all phases	All external attacks, and the following internal attacks Spoofing Redirection by modifying route sequence number	Requires certificate authority or key sharing mechanism
Trust-level metric	All attacks prevented by authentication All attacks on higher trust-level nodes	Requires certificate authority or key sharing mechanism Difficulty to define trust level
Secure neighbor verification	All attacks prevented by authentication Rushing	Requires certificate authority or key sharing mechanism Important overhead when mobility increases
Randomize message forwarding	Rushing	Latency
Onion encryption	All external attacks, and the following internal attacks Spoofing DoS by modifying source route	Requires certificate authority or key sharing mechanism High computational cost

Table 2. *Solutions to secure routing protocols.*

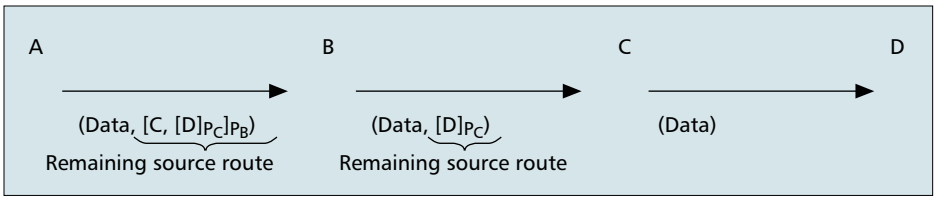

Figure 3. *Example of forwarding a packet using Onion-encryption.*

archal key sharing. The advantage of this solution compared with the previous one is that it prevents attacks from an internal node on a higher trust level.

Secure Neighbor Verification

This solution consists of a three-round authenticated message exchange between two nodes before each one claims the other as neighbor. If this exchange fails, then the well-behaving node ignores the other, and does not handle packets sent by it. This solution beats the illegal use of a high power range to launch the rushing attacks. Since the sender using higher powers cannot receive the packet from further nodes, it will not be able to perform the neighbor detection process, and it will be ignored [21]. The major drawback of this solution is the important overhead when the mobility increases.

Randomize Message Forwarding

This technique is proposed by Yi *et al.* [21] to minimize the chance that a rushing adversary can dominate all returned routes. In traditional RREQ forwarding, the receiving node immediately forwards the first RREQ and discards all subsequent RREQs. Using this scheme, a node first collects a number of RREQs, and selects a RREQ at *random* to forward. There are thus two parameters related to this technique: first, the number of RREQ packets to be collected, and second, the algorithm by which timeouts are chosen. We think the drawback of this solution is that it increases the delay of the route discovery, since each node must wait for a timeout or to receive a given number of packets before forwarding the RREQ. Moreover, the random selection prevents the discovery of optimal routes. Route optimality may be defined as hop number, energy efficiency [28], or another metric, but it is not random.

Onion Routing

Awerbuch *et al.* [27] propose the use of an efficient asymmetric encryption strategy to protect and ensure anonymity for source routes when employing a source routing protocol. This strategy consists of encrypting a discovered source route during route discovery in an *onion-like* form, and transmitting data packets using this onion encrypted route. During the route reply (respectively request broadcasting) phase, each node adds its address to the next (respectively previous) portion of the discovered route, and encrypts the outcome using the public key of the previous node (respectively its own public key). In this manner each node will be able to only read the next hop when data packets are transmitted, and not any other. The onion encryption of a discovered source route (n_0, n_1, …, n_k) is performed during the reply phase as follows:

n_k ID is encrypted with $P_{n_{k-1}}$ (the public key of node n_{k-1}), the result is denoted by $[n_k]_{P_{n_{k-1}}}$; at n_{k-1} this outcome is concatenated to n_{k-1} ID and encrypted with $P_{n_{k-2}}$: $[n_{k-1}, [n_k]_{P_{n_{k-1}}}]_{P_{n_{k-2}}}$, and so on until reaching the source's successor (n_1). The outcome of all these operations is the following onion encrypted source route: $[n_1, \ldots, [n_{k-1}, [n_k]_{P_{n_{k-1}}}]_{P_{n_{k-2}}} \ldots]_{P_{n_0}}$.

This encrypted route will be used to route each data packet. Node n_0 decrypts the route and gets n_1 address, to which it transmits the packet; the remaining part is encrypted with p_{n_1} and cannot be deciphered by n_0. n_1 does the same thing and routes the packet, and so on until reaching the final destination.

Example — Assume a discovered source route (B, C, D), which connects A to D, is to be used by A to transmit a data packet. The onion-encrypted sequence of this route is: $[B, [C, [D]_{P_C}]_{P_B}]_{P_A}$. When decrypting the route with its own private key, node A retrieves B's address, to which it transmits the packet. The other addresses (C, D) are hidden to A, and can not be deducted since they are asymmetrically encrypted (assuming the asymmetric encryption mechanism is robust). Identically, B (respectively C) gets C's (respectively D's) address, using its own private key, to which it forwards the packet. Figure 3 illustrates this example.

This mechanism ensures that each node is only able to identify its successor, where the rest of the route is kept anonymous. Consequently, DoS attack by modifying source route is prevented. When combined with authentication, this mechanism is powerful and efficient, but it suffers from high computation cost.

For each solution presented previously, Table 2 summarizes the preventable attacks and the shortcomings. As illustrated, all the solutions overcome all external attacks, and some internal attacks. Still, almost all the solutions require a certificate authority or a key sharing mechanism, which is problematic in MANETs, as we will see later. Randomized message forwarding is the only solution that is not based on such a requirement, but the problem with this approach is the important latency it introduces. Each solution has other drawbacks, as discussed previously.

Data Forwarding Security Issues

Protecting the network layer in MANETs is a highly important research topic. The core functionalities provided in this layer are routing and packet forwarding, and are closely related. The data forwarding service consists of *correctly* relaying the received packets from node to node until they reach their final destination, following the routes selected and maintained by the routing protocol. Malicious attacks or selfish misbehavior on either of them will disrupt the normal network operations. In the previous section we focused on the routing protocol itself, and we presented attacks that target routing procedures along with the appropriate solutions. In this section we will deal with the issues related to the packet forwarding service. As in the previous section, we first present threats, then we discuss current solutions.

Data Forwarding Threats

Eavesdropping (Passive Attacks)

The wireless channels used in MANETs are freely and easily accessible. Moreover, the promiscuous mode, which means capturing packets by a node that is not the appropriate destination, is allowed and employed by protocols to operate or to ensure more efficiency, e.g. a routing protocol may use this mode to learn routes. These features can be exploited by malicious nodes to eavesdrop packets in transit, then analyze them to obtain confidential and sensitive information. The obvious preventive solution to protect information is to encrypt packets, but data encryption does not prevent malicious nodes from eavesdropping and trying to break decryption keys. To the best of our knowledge no detecting solution

is currently available, and detecting this attack is a largely open research topic. Since breaking keys is always possible and key revocation in MANETs is problematic as we will see later, eavesdropping remains a serious attack against data forwarding.

Dropping Data Packets Attack

Since packets follow multi-hop routes and pass through mobile nodes, a malicious node can participate in routing, include itself in routes, and drop all packets it gets to forward. To do this, the malicious node first attacks the routing protocol to gain participation in the routing, using one or more of the attacks presented previously. This attack has the same effects as the selfish misbehavior presented hereafter and the same solutions may be applied.

Selfish Behavior on Data Forwarding

In many civilian applications, such as networks of cars and the provision of communication facilities in remote areas, the nodes typically do not belong to a single authority and do not pursue a common goal. In such networks, forwarding packets for others is not in the direct interest of nodes, so there is no good reason to trust nodes and assume that they always cooperate. Indeed, a selfish node may try to preserve its resources, particularly battery power, which is a precious resource, by dropping packets it is asked to forward while using other nodes' services and consuming their resources to transmit its own packets toward remote nodes. This is not an intentional attack but a selfish misbehavior. However, it represents a potential danger that threatens the quality of service in the network as well as one of the most important network security requirements, i.e. availability (discussed in an earlier section). Hereafter we present the detection and preventive solutions currently proposed against this misbehavior.

Detection Solutions Against Selfishness on Data Forwarding

End to End Feedbacks

This mechanism consists of acknowledging packets on the network layer in an end-to-end manner, to render the routing protocol reliable (like TCP). That is, the destination node acknowledges the successfully received packets by sending feedbacks (ACKs) to the source. A successful reception implies that the corresponding route is operational, while a failure in the ACK reception after a timeout may be considered as an indication that the route is either broken, compromised, or includes selfish nodes. Routing protocols based on this approach maintain a rating for each route; this rating reflects the route's reliability and is updated each time a piece of data (a set of data packets) is transmitted across the route. It is increased for each successful reception (when the source receives the ACK of that piece), and decreased for each failed piece (when a timeout expires without receiving an ACK); lost packets may be retransmitted in this case. When the path rating of a given route decreases below a defined *threshold*, which is high enough to overcome the losses due to collisions, this route will not be used anymore. Moreover, the routing protocol may be changed to rely on this rating as a metric and choose the most reliable routes.

The major problem with this technique is the lack of misbehaving node detection. This technique may detect both routes containing misbehaving or malicious nodes, and those that are broken, but without making any distinction between these two cases and without any further information regarding the node causing the packet loss. However, this technique helps to avoid useless packet transmission through unreliable routes, and it can be combined with other more sophisticated techniques. It is used in SMTP [29] along with another technique, called data dispersal, which will be presented later. It is also used in [27] combined with probing, which also will be presented later.

Monitoring in the Promiscuous Mode (Watchdog)

To the best of our knowledge, Marti *et al.* were the first to have addressed the problem of node misbehavior in data forwarding in MANETs. In [30] they defined the watchdog method, a basic technique on which many further solutions rely. It aims to detect misbehaving nodes that do not forward packets (or malicious nodes that intentionally drop packets) when using a source routing protocol, by monitoring neighbors in the promiscuous mode.

Suppose node S sends packets to D using a route including (among others) three intermediate nodes, A, B, and C. When A transmits a packet to B to forward to C, A can check if B forwards each packet by analyzing packets it overhears during a given timeout. If A overhears a packet it is monitoring during the fixed timeout, then it validates its forwarding, otherwise it raises a rating regarding B, and will judge that B is misbehaving when the rate exceeds a given threshold, then notifies S. This monitoring is generalized for each pair of hops in the source route.

The watchdog is able to detect misbehaving nodes in many cases, and requires no overhead when no node misbehaves. Nevertheless, it fails to detected the misbehavior in some cases. For instance, after a collision at C, B could circumvent retransmitting the packet without being detected by A. The watchdog also fails when two successive nodes collude to conceal the misbehavior of each other, that is B could collude with C and not report to A when C misbehaves. However, this cooperative misbehavior is very hard to detect. Another drawback with this solution is that it may cause false detections, especially when the monitored node uses the power control technique [31] to preserve its power. When C is closer to B than A, and when B transmits packets using controlled power according to the distance separating it from C, A could not overhear B's forwarding, and may accuse it wrongly. Note that the power control technique has been used by many routing protocols proposed after the watchdog's proposal in the field of power consumption optimization, e.g. [28, 31, 32].

Activity-Based Overhearing

Kargi *et al.* [33] propose the so called *activity-based overhearing* technique, which is a generalization of the watchdog technique. In this technique, a node constantly monitors in the promiscuous mode the traffic activity of all its neighbors for regular data packets, and oversees the forwarding of each packet whose next forwarder is also in its neighborhood. This can increase the number of observations and improve the watchdog efficiency. This general technique also suffers from all the problems cited above, especially that related to the power control technique, since it relies on promiscuous mode monitoring.

Mutually According Admission in Neighborhood

Yang *et al.* [34] have described a unified network layer solution based on the approach of mutually according admission in neighborhood. This technique aims at protecting both routing and data forwarding. A threshold cryptography-based signature [35] and the watchdog technique [14] are at the core of this solution. In the following we briefly describe this approach:

Nodes in a neighborhood mutually accord participation admissions, and nodes without *up-to-date* admissions are

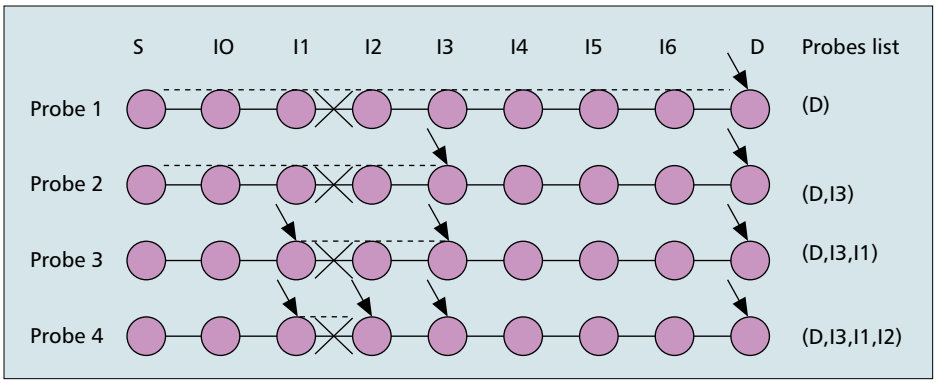

Figure 4. *Example of the basic probing.*

excluded from any network service. Each node has a token issued by its local neighbors, which allows it to participate in the network operations. The token has a period of expiration, whose value depends on how long the holder node has been behaving well. This latter renews (updates) the token before its expiration. Nodes in a neighborhood collaboratively monitor each other to detect any misbehavior.

There is a key pair SK/PK (secret key and public key); each token carried by a node is signed with the global secret key SK, and is broadcast periodically in the hello message to ask for a new validation. The token validation can be checked using the PK at any node. Note that the hello message or the beacon is a special message periodically broadcast by each node to inform nodes in a neighborhood about the presence of the broadcaster. Each node has a partial key that is a part of the SK and participates by providing a *partial signature of order K*; thus *K* different partial signatures are sufficient to provide the right signature. In other words, the SK is divided among nodes in such a way that *K* different signatures with *k* different partial keys are necessary and sufficient to make a signature equivalent to that made by the SK. This technique is called polynomial secret sharing [35, 36]. To decide whether or not to provide a partial signed token for the requestor, the requestor's historical behavior is considered. For this purpose, the watchdog is employed to monitor neighbors and detect any misbehavior, and requests from nodes considered misbehaving are denied.

A node has to have at least *k* neighbors, otherwise it cannot get a valid signed token, and will consequently be unfairly excluded. Moreover, all the watchdog's detection problems remain unresolved with this solution, since the watchdog is used for the monitoring.

Reputation-based Solutions

Reputation is the amount of trust inspired by a particular member of a community in a specific setting or domain of interest [37]. Members that helpfully contribute to the community life develop a good reputation among the community's members, while others who refuse to cooperate are badly reputed and gradually excluded from the community. Reputation systems have been proposed for a variety of applications, among them we cite the selection of good peers in a peer-to-peer network, the choice of transaction partners for online auctioning, and especially the defense against misbehaving and malicious nodes in mobile ad-hoc networks [38].

In our context, the reputation of a node is the amount of trust the other nodes grant to it regarding its cooperation and participation in forwarding packets. Hence, each node keeps track of each other's reputation according to the behavior it observes, and the reputation information may be exchanged between nodes to help each other to infer the accurate values. There is a trade-off between efficiency in using available information and robustness against misinformation. If ratings made by others are considered, the reputation system can be vulnerable to false accusations or false praise. However, if only one's own experience is considered, the potential for learning from the experiences of others goes unused, which decreases efficiency.

Two solutions belonging to the reputation-based category have been proposed, namely CORE [37] and CONFIDANT [39, 40]. In the first solution, positive observations (of well-behaving nodes) are propagated but not the negative observations, which reduces the potential for learning from the observations made by others and can decrease the efficiency of misbehavior detection in the network. On the other hand, CONFIDANT's reputation system is built on negative experiences rather than positive impressions. Buchegger *et al.* [38] justified this by the fact that misbehaving is ideally the exception rather than the norm. To mitigate the vulnerability to DoS attacks by propagating false accusations, CONFIDANT's trust manager uses a rate function that assigns different weights to the types of behavior detections, in such a way to give more importance to local observations when computing reputation rating. Both CONFIDANT and CORE use monitor compounds that fully rely on the promiscuous mode monitoring; they consequently inherit all the watchdog's detection drawbacks.

Probing

This mechanism consists of simply incorporating commands into data packets to acknowledge them; these commands are called probes and intended for selected nodes. Probes are generally launched when a route that contains a selfish or malicious node is detected (but not the ID of that node). Awerbuch *et al.* [27] are the first to use this mechanism. The protocol they have proposed is based on end-to-end feedbacks, and thus requires the destination to return an ACK (acknowledgment) to the source for every successfully received data packet. The source keeps track of the number of recent losses (ACKs not received over a window of recent packets). If the number of recent losses violates the acceptable threshold, the protocol registers a fault between the source and the destination and starts a dichotomic search on the path in order to identify the faulty link. The source controls the search by specifying a list of intermediate nodes on future data packets; identifiers of these nodes are onion-encrypted. Each node in the list, in addition to the destination, must send an ACK for the packet; these nodes are called probed nodes. The list of probes defines a set of non-overlapping intervals that cover the whole path, where each interval covers the subpath between the two consecutive probes that form its endpoints. When a fault is detected on an interval, the interval is divided into two by inserting a new probe. This new probe is added to the list of probes appended to future data packets. The process of sub-division continues until a fault is detected on an interval that corresponds to a single link, as shown in Fig. 4. In this example node I_2 is assumed to be a selfish node that drops all packets, including those containing probing; it is also assumed not to be replying to probing commands. As illustrated, I_2 will be detected after four probes (when the previous assumptions are held).

The major drawback of this solution is that a misbehaving node can easily realize a probing when launched; it only needs to analyze packets it receives before dropping them, and therefore it can circumvent the probing. In the context of the

previous example, node I_2 could forward packets, including probing and reply packets, to the probing sent to it. In this manner no failure probing will take place,[1] and thus this probing will not be able to detect the selfish node in this case.

A more enhanced solution has been proposed by Kargl *et al.* [41]. This solution uses *iterative probing*, which defers from the previous solution in the fact that each command is addressed to one node instead of a set of nodes, therefore the command contains one encrypted node ID added to a special field in data packets. If a data packet includes no probing command, then the field will contain a random number, such that a recipient cannot distinguish data packets including probing from regular data packets, unless it is the destination of the probing command. This solution is also unreliable, as it makes possible the detection of the link containing the selfish (or the malicious) node, but cannot distinguish which of the two nodes forming the link is actually the misbehaving one, since there is no knowledge of the selfish node behavior upon the reception of a probing (either it sends back the ACK or not). In the previous example, the iterative probing detects the link (I_1, I_2) but cannot detect the appropriate misbehaving. To mitigate this problem Kargl *et al.* proposed *unambiguous probing*. The principal of this mechanism is simple and can be summarized as follows. After an iterative probing, a link (I_i, I_{i+1}) will be detected. To determine which one of the two suspicious nodes is the guilty (the misbehaving) node, the source node asks node I_{i-1} to check if it can *promiscuously overhear* the forwarding of I_i. If so, I_{i+1} is the guilty node, otherwise the guilty node is I_i. We think this mechanism (unambiguous probing) suffers from the watchdog's problems, since it relays on promiscuous monitoring at the predecessor of the suspicious link.

Preventive Solutions Against Selfishness on Data Forwarding

Thus far, we have presented detective solutions that aim at detecting the selfish misbehavior (or the attack) on packet forwarding when it appears in the network. Another approach is to *proactively* try to avoid this misbehavior, either by motivating nodes to cooperate or by taking measures to prevent packets from being dropped before sending them. In the following sections we discuss two solutions we classify as preventive.

Nuglets

In the framework of the Terminodes Project [42], Buttyan and Hubaux [43, 44] have proposed an economic-based approach that stimulates the nodes to cooperate. They have modeled and analyzed this approach in their further work [45]. The authors have introduced what they call virtual currency or nuglets, along with mechanisms for charging/rewarding service usage/provision. The main idea of this technique is that nodes which use a service must pay for it (in nuglets) to nodes that provide the service. This makes nuglets indispensable for using the network, and motivates each node to increase its stock of nuglets by providing services to other nodes. Besides the stimulation for the provision of services, this mechanism can also force the nodes to make moderate usage of the network services that become charged. This approach is a general proposal that can be applied to any service. The authors applied it to the forwarding service.

The first question behind this proposal that has to be answered is how to represent nuglets. The authors suggest presenting them by counters in the nodes; each node has a nuglet counter whose value corresponds to the node's wealth. However, in order to prevent the node from illegitimately increasing its own counter, the counter is required to be maintained by a trusted and tamper-resistant hardware security module in each node, on which the robustness of this solution relies. This complicates the implementation and may represent a disadvantage.

In this solution, if a node behaves selfishly it cannot earn nuglets and will be unable to send its own packets. Moreover, this solution allows the node's redemption, since a node that is unable to send its own packet because it runs out of nuglets is not excluded from being asked to participate in the data forwarding service; hence, it can always forward packets and earn nuglets. However, an important drawback of this approach is that a well-behaved node that is not asked to route enough packets, due to its position and/or the communications of its neighbors, could not earn nuglets and will be unable to send its own packets.

Data Dispersal

This approach is based on Rabin's algorithm [11], which takes advantage of the existence of multiple routes from a source to a destination to increase reliability when transmitting packets. It consists of adding *redundancy* to the message to send, then the message and the redundancy are encoded and divided into a number of pieces and dispersed on the available routes, so that even a *partial* reception can lead to the successful reconstruction of the message at the receiver. Note that node-disjoint routes ensure more efficiency, since each node misbehaving affects solely a single route when using these types of routes. This technique can mitigate partial packet loss that can occur due to misbehavior on some used routes, since the encoding and the dispersion allows the reconstruction of the original message with successful reception of any M out of N transmitted pieces.[2]

The ratio N/M, or *redundancy factor,* is a crucial parameter for this solution. Increasing this ratio ensures more reliability, since a fewer number of pieces among the overall sent pieces would be required to reconstruct B, but this choice causes an important redundancy. On the other hand, decreasing the redundancy factor reduces the redundancy, but provides less reliability. Hence, the choice of this parameter is a challenging issue, and should strike a balance between reliability and redundancy.

Even though this mechanism does not prevent nodes from misbehaving and does not motivate nodes to cooperate, unlike the previous approach, we think it is helpful and can reduce selfish misbehavior and the effects of attacks on the communication reliability, and can be combined with a reactive solution. Papadimitratos and J. Hass [29] have proposed SMTP, a solution that uses this mechanism. But this solution has the drawbacks of the end-to-end feedback technique described before, since it relies on it. Table 3 summarizes the main features and drawbacks of all the detection and preventive solutions presented in this section.

MAC Protocol Security Issues

After discussing the security issues related to the network layer in the two previous sections, we now discuss MAC protocols. We present a misbehaving activity that threatens one

[1] *No faulty internal will be detected.*

[2] *A detailed description of message encoding and decoding can be found in [11]; it has been omitted since it requires a rigorous mathematical presentation.*

Solutions	Class	Features	Drawbacks
End-to-end feedbacks	Detective	The destination sends back ACKs to the source Detects routes that include misbehaving	Does not detect the appropriate node Important overhead
Watchdog	Detective	Promiscuous mode usage Detects misbehaving in many cases No overhead when there is no misbehaving	Fails to detect the misbehaving in the following cases: • after a collusion • cooperated misbehavior • when the monitored control its transmission power Causes faults detection when using adaptable transmission powers
Activity-based overhearing	Detective	Generalization of the watchdog Provides more traffic to monitor More efficiency when the watchdog is operational	Watchdog's drawbacks
Mutually according admission	Detective	Nodes in neighborhood accord to each other participation admission Uses threshold cryptography	Unfairly excludes nodes with less than the predefined threshold (K) from the service Watchdog's drawbacks
CORE	Detective	Based on reputation Only positive impressions are propagated	No propagation of misbehaving detection Watchdog's drawbacks
CONFIDANT	Detective	Based on reputation Built on negative impression	Watchdog's drawbacks
Dichotomic probing	Detective	Incorporates commands into data packets Requires end-to-end feedback employment Uses onion encryption	Temperable
Iterative probing	Detective	Incorporates commands into data packets Requires end-to-end feedback employment Uses asymmetric encryption Detects the link containing the misbehaving node	Fails to detect the appropriate misbehavior
Unambiguous probing	Detective	Iterative probing + watchdog employment on the detected link upstream node	Watchdog's drawbacks
Nuglets	Preventive	Economic-based approach Nodes that use the service pay nodes that offer it Motivates nodes to cooperate Forces nodes to make moderate usage of the service Allows nodes' redemption	Unfairly prevent nodes not asked to forward packets from sending their own packets
Data dispersal	Preventive	Based on Rabin's algorithm Requires multi-path routing Reduces the misbehaving effects and increases the reliability	Does neither detect nor prevent nodes misbehavior

Table 3. *Solutions to secure data forwarding.*

of the most important purposes of MAC protocols, namely fairness in channel access.

Misbehaving in Channel Access

The Problem

Since there is no central authority in MANETs, Wireless Medium Access Control (MAC) protocols [46], such as IEEE 802.11, use distributed contention resolution mechanisms for sharing the wireless channel. The contention resolution is typically based on cooperative mechanisms that ensure a reasonably fair share of the channel for all the participating nodes. In this environment, some selfish hosts in the network may misbehave by failing to adhere to the MAC protocol, with the intent of obtaining an unfair share of the channel. The presence of selfish nodes that deviate from the contention resolution protocol can reduce the throughput share received by conforming nodes.

The IEEE 802.11 MAC protocol [47], which is the standard MAC protocol for wireless networks, has two

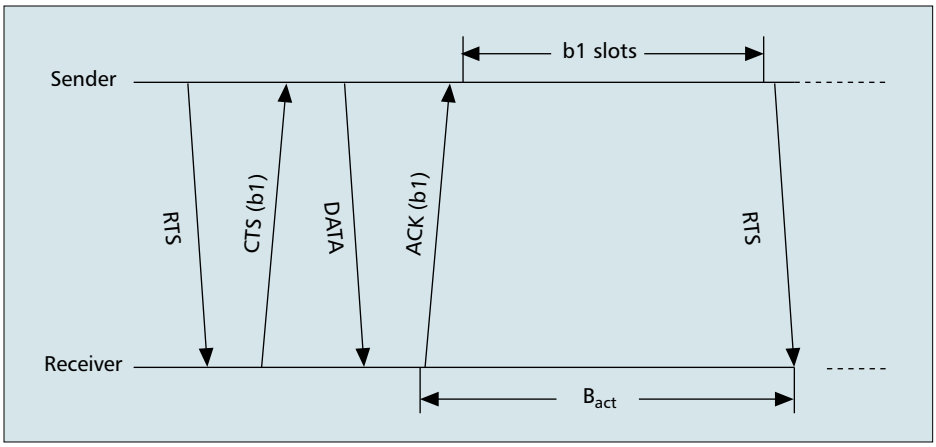

Figure 5. *Receiver-sender interaction.*

mechanisms for contention resolution: a centralized mechanism called PCF (Point Coordination Function), and a fully distributed mechanism called DCF (Distributed Coordination Function). PCF needs a centralized controller (such as a base station) and can only be used in infrastructure-based networks; thus, it is not to be considered in the ad hoc mode. In contrast, DCF is widely used in infrastructure-based wireless networks as well as in ad hoc wireless networks.

DCF uses the CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) option for resolving contention among multiple nodes accessing the channel. A node (sender) with data to transmit on the channel selects a *random backoff* value from range [0;CW], where CW (contention window) is a variable maintained by each node. While the channel is idle, the backoff counter is decremented by one after every time slot (a fixed interval of time), and the counter is frozen when the channel becomes busy. The node may access the channel when the backoff counter is decremented to zero. After the backoff counter is decremented to zero, the sender may reserve the channel for the duration of the data transfer by exchanging control packets on the channel. The sender first sends a RTS (request to send) packet to the receiver, then the receiver responds with a CTS (clear to send) packet. This RTS-CTS exchange is optional in IEEE 802.11; it aims to ensure the channel reservation for the duration of the data transmission. Both of the packets contain the proposed duration of the data transmission. Other nodes that overhear either the RTS or the CTS (or both) are required to defer transmissions on the channel for the duration specified in the RTS/CTS. After a successful RTS/CTS exchange, the sender transmits a DATA packet, which will be acknowledged by an ACK. If the node's data transmission is successful, the node resets its CW to a minimum value (CWmin); otherwise, if the sender does not receive the CTS, then CW is doubled, but it should not exceed a maximum value of CWmax. A misbehaving node may obtain more than its fair share of the bandwidth by:

- Selecting backoff values from a different distribution with smaller average backoff value than the distribution specified by DCF (e.g., by selecting backoff values from the range [0,CW/4] instead of the range [0,CW]) [48].
- Using a different retransmission strategy that does not double the CW value after collisions.

We note that it is not beneficial for a selfish node to not delay at all or to choose a very small constant period, since this may result in a very high collision rate, and thus the loss of the packets it sends.

Such selfish misbehavior can seriously degrade the throughput of well-behaved nodes. For instance, simulation results obtained by Kyasanur and Vaidya[48] show that for a network containing eight nodes sending packets to a common receiver with one of the eight nodes misbehaving by selecting backoff values from the range [0,CW/4], the throughput of the other seven nodes is degraded by as much as 50 percent.

To the best of our knowledge there is no published solution proposed to this complex problem, except the solution proposed by Kyasanur and Vaidya [48]. In the following section we present and discuss this solution.

The Solution

Due to the random selection of the backoff, it is hard to distinguish between the legitimate selection of small backoff values and a misbehaving selection. Hence, detecting a MAC layer misbehavior is a complicated problem. Kyasanur and Vaidya [48] have proposed a scheme to resolve this problem.

The sheme consists of modifications to the IEEE 802.11 protocol that enable a receiver to identify sender misbehavior within a small observation interval. Instead of the sender, the receiver selects a random backoff value, b1, and appends it to the CTS and the ACK packets it transmits to the sender. The sender uses this assigned backoff value in the next transmission to the receiver. This exchange is summarized in Fig. 5. With these modifications, a receiver can identify senders deviating from the protocol by observing the number of idle slots between consecutive transmissions from the sender (B_{act} in Fig. 5). If this observed number of idle slots is less than the assigned backoff, then the monitor realizes that the sender may have deviated from the protocol. The magnitude of observed deviations over a small history of received packets is used to diagnose sender misbehavior with high probability. The proposed scheme also attempts to negate any throughput advantage that the misbehaving nodes may obtain. To achieve this and discourage misbehavior, deviating senders are penalized, i.e., when the receiver perceives a sender to have waited for less than the assigned backoff, it adds a penalty to the next backoff assigned to that sender.

Discussion

If we assume *the receiver is a well behaving* node, when the sender does not backoff for the duration specified by the penalty, it significantly increases the probability that it will be detected as a misbehaving node. Moreover, the punishment applied to the detected misbehaving node is a disincentive to such nodes to behave in this way. However, there are some problems with this solution:

- Deviations observed are not inevitably a misbehavior; they may be caused by the channel condition difference between the sender and the receiver, or what is known as the hidden terminal problem [46]. To illustrate this, we give the following example. Consider the situation of three nodes A, S, and R, where R is the receiver and S is the sender in our context, i.e., R is monitoring S after sending it a backoff. We assume R is within the power range of A, but S is not (Fig. 6). We also assume that A is transmitting packets. Hence, R's channel is busy, whereas S's channel is free. Consequently, S decreases its counter upon each slot unlike R, then the former will be able to access the channel when its backoff counter

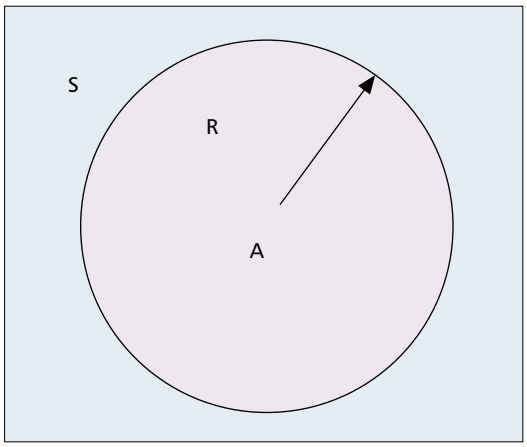

Figure 6. *Nodes' positions.*

reaches 0, and this access will be considered by R as a deviation. If such a situation persists, R will register enough deviations to wrongly consider S to be misbehaving, so this solution may cause *innocent* nodes to be accused of misbehaving (false positives).

- We define a new misbehavior in channel access that we call *cooperative misbehavior*. We consider two nodes that wish to obtain unfairly high throughput to exchange packets, when they are in the same neighborhood (they are neighbors). The receiver may misbehave and assign unfair backoff values to the corresponding sender. The proposed scheme fails with this situation since it relies on the receivers' trustworthiness.

Key Management

As we have seen earlier, most of the solutions proposed for securing routing and data forwarding rely on cryptography, and assume the existence of an underlying mechanism for providing and managing keys. Many secure applications and services also use cryptography and rely on this assumption. However, because of the lack of any central infrastructure or administration, key management is problematic in MANETs.

There are basically two kinds of key infrastructure. The first involves the private key infrastructure, which establishes common private keys used for symmetric cryptography, such as symmetric group keys used for securing group communications[3] [49–53]. The second kind is the public key infrastructure, which provides a couple of keys (public/private) used for asymmetric cryptography, as in digital signatures. Providing such an infrastructure in MANETs is challenging, due to their infrastructureless nature. Indeed, the role of this infrastructure should be spread out to all mobile nodes (or a subset of them), which form the key infrastructure. Therefore, the MANETs' key management system should neither trust nor rely on any fixed certificate authority (CA), but should be distributed and self-organized.

Private Key Infrastructure

The private key management protocols have been classified into two classes [54]: key distribution protocols, which are centralized and based on a trusted third party, and key agreement protocols, which are distributed. The suitable class for our environment is certainly the second approach. Hereafter, we focus our discussion on solutions belonging to this class. We will deal with some novel solutions specially devoted to MANETs, and also with some previous solutions that were either directly adopted in MANETs or used to build new sophisticated solutions.

Diffie-Hellman Two-party Agreement (DH)

This basic protocol, proposed in a landmark paper [55], allows two nodes to build a common key. The principal of this protocol is simple: the two involved nodes, M_1 and M_2, send one another a partial key to be used for the common key computation. M_1 generates a random number r_1 ($1 \le r_1 \le p$), and sends α^{r_1} to M_2, such that α and p are constants known by each node. On the other hand, M_2 generates a random number r_2, and sends α^{r_2} to M_1. Thereby, each node could compute the common key, which is $\alpha^{r_1 \times r_2}$ This solution is based on discrete logarithmic arithmetic, and also relies on the agreement on the parameters α and p between the two nodes. Although it is simple and limited to two nodes' common key establishment, this protocol was used to design more sophisticated protocols, as we will see later.

General Diffie-Hellman (GDH)

Steiner *et al.* [56] proposed a n-party generalization of the basic two-party DH protocol (described before). The new protocol consists of n rounds, allowing n nodes to establish a common key. In the first $n - 1$ rounds contributions are collected from each node. In the first round, M_1 generates r_1 and computes α^{r_1}, which it sends to M_2. In the second step M_2 generates r_2, computes α^{r_2} and sends it to M_3, along with α^{r1} and $\alpha^{r_1 \times r_2}$. This latter sends to M_4 (after making the required computations) the third-round partial factors, i.e., $\alpha^{r_1 \times r_2}$, $\alpha^{r_1 \times r_3}$, $\alpha^{r_2 \times r_3}$, as well as the third-round partial key $\alpha^{r_1 \times r_2 \times r_3}$. This process continues for each M_i ($i < n$). Upon the $(n - 1)$th round, the collector node M_n receives the $(n - 1)$th round partial factors, and the $(n - 1)$th round partial key, then it generates its random number and computes the final key K.[4] In the last round, node M_n sends each M_i the appropriate (n)th round partial factor, i.e.,

$$\alpha^{(\prod_{j=1}^{n} r_j)/r_i}.$$

Consequently, each node uses its random number to compute the common key K. Note that partial factors are used to avoid sending the final resulted key during the last round. Also note that the $(n - 1)$th round requires $n - 1$ operations (sending the partial factor to each node), which makes the computational complexity of the solution 0 ($2 \times (n - 1)$), as shown in Table 4. Figure 7 is an illustrative example of this protocol.

Even though it uses a collector, this solution is contributory, since each node contributes to the key computation with the random number it generates. Nevertheless, the major drawback of this solution is the important overhead, due to the message size rising from round to round.[5] This can also cause a problem with scalability.

Password Authentication-based Key Exchange

Another approach proposed for distributed systems and largely adopted for MANETs is password authentication. The aim of this technique is to build a strong private key from a weak share. To this end, Bellovin and Merrit [57] have proposed a protocol called "Encrypted Key Exchange" (EKE), in which

[3] *Public keys also might be used in group communications, especially when using digital signature.*

[4] $K = \alpha^{(\prod_{j=1}^{n} r_j)}$

[5] *Particulary, the partial factors.*

Solutions	Distribution	Based on secret share	Contributory	Computational complexity
GDH	Partially (uses collectors)	No	Yes	$2 \times (n - 1)$
n-party Pwd authentication	Partially	Yes	No	$6 \times (n - 1)$
Contributory n-party Pwd authentication	Partially	Yes	Yes	$4 \times (n - 1)$
Hyper-cube	Totally	No	Yes	$2 \times \log(n)$
Octopus	Almost totally	No	Yes	$2 \times (\log(n) + 1)$
Hyper cube + Pwd	Totally	Yes	Yes	$4 \times \log(n)$
GDH + Pwd	Partially	Yes	Yes	$4 \times (n - 1)$
Cluster-based	Partially (only between leaders)	No	Yes	

Table 4. *Private key management solutions.*

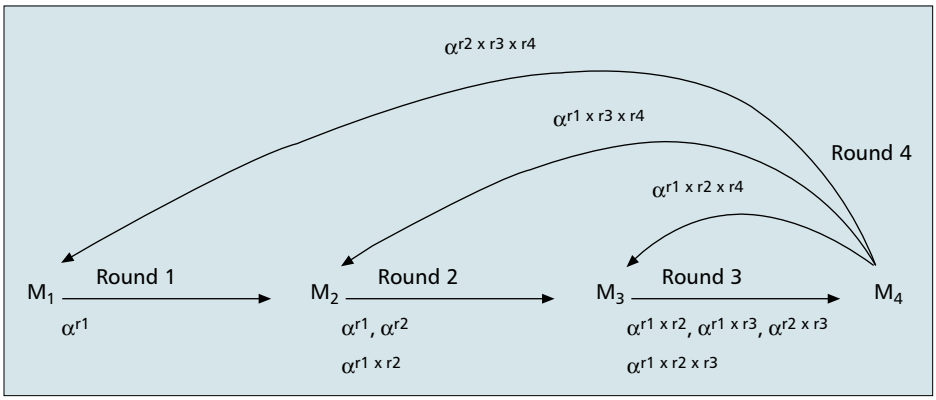

Figure 7. *Example of General Diffie-Hellman with four nodes.*

two parties could derive a strong key from a weak shared password. Unlike the password, the constructed key is not vulnerable to dictionary attacks, which makes it usable for a long period.

The authors assumed that the involved nodes, M_1 and M_2, share a weak secret password P, and use a public function as well as a one-way function(f (., .)). M_1 is also assumed to possess a random key pair for encryption and decryption, respectively. The two nodes establish a strong key after a five-round message exchange.

During these rounds each node M_i generates a random string (S_{Mi}) it confidentially exchanges with the other. Using the public function and the random keys, M_i ensures authentication of M_2 in the third step, i.e., M_i ensures that M_2 is a node that knows p. This latter ensures that M_1 knows p upon the fourth step, using the previous functions and a random key it generates during the first steps. At that point, each node computes the common key:

$$K = f(S_{M2}, S_{M2}) .$$

More details of the algorithm and its steps can be found in [57].

One obvious proposed way to extend this protocol to multiple parties is to elect a leader, and to execute the previous two-party protocol between each node and the leader. At the end of the protocol run, each node shares a key with the leader. An additional step will be required for this latter to pick a common key, and to distribute it to all the parties using the pairwise session keys it shares with them. Overall, the computation complexity of this n-party solution is $O6 \times (n - 1)$; $5 \times (n - 1)$ for the execution of the five-step two-party protocol between each node and the leader; and $(n - 1)$ for the additional step. One disadvantage of this solution is its overhead, since each step consists of one-to-many (from M_n to each M_i) or many-to-one (from each M_i to M_n) message exchange, which is very expensive in a multi-hop environment. Another drawback of this solution is that the common session key choice is not contributory but unilateral, viz. unlike the general DiffieHellman protocol described before, the leader selects by itself the common key, thus the key is not inevitably composed of contributions of other parties. The only contribution of these nodes is for the pairwise session key construction. To mitigate this problem Asokan and Ginzboorg [58] have proposed a contributory multi-party version, which we will discuss in the next section.

Contributory Password Authentication

Asokan and Ginzboorg [58] have proposed a contributory extension of the previous solution. The new protocol also relies on a collector node, and is composed of the following four steps:

1 The collector M_n sends to all nodes its public key E[6] encrypted with the shared secret P.
2 Each node M sends M_n two random quantities S_i and R_i, double-encrypted with E and P. At this point, M_n can ensure that each party knows P.
3 M_n sends each M_i the set $\{S_j, j = 1...n\}$ encrypted with R_i.
4 Finally, each M_i computes the private key $K = f(S_1, \ldots, S_n)$, where $f()$ is an n-input one-way function. Each M_i sends K (S_i, $H(S1, S2, \ldots Sn)$) to M_n for key confirmation, such that $h()$ is a public hash function.

[6] *This key is just temporary used for the private key establishment, it can be generated randomly by the node.*

In this solution, each node M_i participates in the construction of K with S_i, which makes the solution contributory. This solution requires only four steps instead of six, which decreases its latency and improves its computational complexity compared with the previous solution (as shown in Table 4). However, two-password authentication-based solutions require a pre-share of a secret, which is not always possible in dynamic networks.

Hyper-Cube-based Approach

Another extension of the Diffie-Hellman (DH) protocol to n-party adopted for MANETs has been proposed by Becker and Wille [59].[7] In this fully distributed approach, nodes are arranged in a d dimensional hypercube, and d fulfills the following condition: 2^d = nodes number. To clarify the approach we first describe it for $d = 2$, i.e., nodes number $n = 4$, then we will generalize this description.

For four nodes $M_{1...4}$, the protocol runs two rounds (steps). In the first round M_1 and M_2 engage a two-party DH protocol and calculate $S_{1,2} = \alpha^{r_1 \times r_2}$. Similarly and simultaneously, M_3 and M_4 engage a two-party DH protocol and calculate $S_{3,4} = \alpha^{r_3 \times r_4}$. In the next round, M_1 and M_3 participate in a new two-party DH protocol using, respectively, $S_{1,2}$ and $S_{3,4}$ as the random number r.[8] M_2 and M_4 do the same thing simultaneously. The result at the end of the protocol is the computation of $K = \alpha^{r_1 \times r_2 \times r_3 \times r_2}$.

This approach can be easily generalized for $d > 2$ as follows. In each dimension j of the d dimensional hyper-cube, each node has one neighboring partner, with which it performs a two-party DH during the j^{th} round using the outcome of the previous round $(j-1)^{\text{th}}$, as described before. Upon the d^{th} round, each node will share the same key with the others. Figure 8 summarizes the protocol for $d = 3$. The protocol execution takes only d rounds which is $\log_2(n)$, whereas GDH's execution requires n rounds. This decreases the computation complexity from $O(2 \times (n-1))$ to $O(2 \times \log_2(n))$, as shown in Table 4. Moreover, the overhead exchanged by each couple of nodes is limited, and it is fixed and unaffected by the number of steps, unlike in GDH, in which the overhead increases from one step to another. Thanks to this advantage, the hyper-cube based solution might be more scalable than GDH. Nevertheless, the price is that all the nodes are involved in each step.

This solution needs the number of nodes n to have the form $n = 2^d$, otherwise (when $2^d < n < 2^{d+1}$) it does not function. The protocol "octopus" [59] solves this problem as follows. The first 2^d nodes play the role of central controllers. The remaining $n - 2^d$ nodes are distributed among the central controllers in such a way that there will be no more than one node associated to each controller. In a first additional round, controllers simultaneously execute two-party DH with their associate nodes. Afterward, controllers engage in a d round hyper-cube protocol run using the information collected in the first round, resulting in a common key sharing. In a last added round the key is distributed to the associated nodes. Another hyper-cube approach problem remains untreated with "octopus," namely, the nodes' arrangement in the hyper-cube, viz. the construction of partners in each dimension, which is equivalent to the nodes' ordering. If this ordering is predefined and static, then the overhead becomes important, and the message exchange between two partners would not cost inevitably one unit as expected, because partners could not remain neighbors during all the protocol executions (due to node mobility). On the other hand, establishing node ordering dynamically requires important overhead when the nodes' mobility increases.

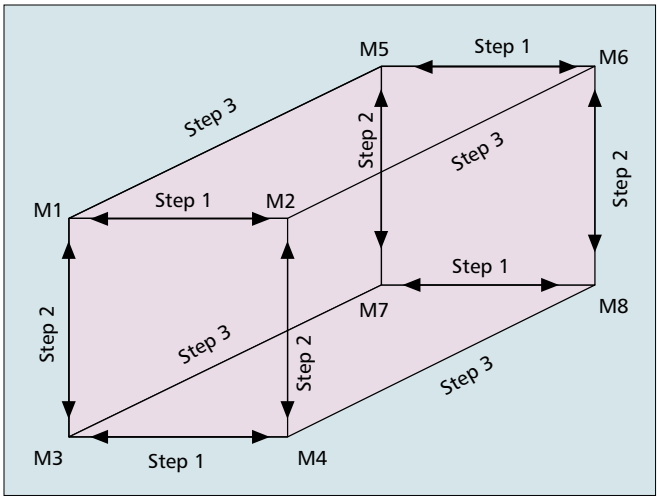

Figure 8. *Hyper-cube protocol.*

Some Hybrid Solutions

Asokan and Ginzboorg [58] proposed a hybrid solution in which they add the password authentication principle to GDH. It is a generalization of the password authentication application to the two-party DH (generalization of authenticated two-party DH) that we describe hereafter.

Assume that the two nodes A and B share a password P. The execution of authenticated two-party DH between these nodes consists of four steps. In the first step, A *encrypts* its contribution (α^{r_1}) with P and sends the outcome to B. This latter does the same thing with its contribution (α^{r_2}) and computes the common key $K = \alpha^{(r_1 \times r_2)}$, and then it sends to A its contribution encrypted with p along with another random number C_b (it generates) encrypted with K. In the third step, A retrieves α^{r_2} (after decryption), computes K that it uses to get C_b, and generates a random number C_a. Afterward it encrypts both C_a and C_b with K and sends them to B. Getting back C_b ensures A's authentication for B. This latter does the same thing and sends back C_a encrypted with K in the final step. At that point, A will be convinced about B's authentication.

The authors proposed to use this basic protocol with GDH as follows. After collecting the $(n-1)^{\text{th}}$ round GDH partial key ($\alpha^{r_1 \times \ldots \times r_{n-1}}$), node M_{n-1} sends each node this key, then each node sends M_n another item, C_i (computed from the partial key and a random number), ciphered with p. This latter sends back to each M_i a combination[9] of C_i and its own contribution α^{r_n}, enabling each node to compute the final key $K = \alpha^{r_1 \times \ldots \times r_n}$. The protocol takes $4 \times (n-1)$ steps: $2 \times (n-1)$ to compute and distribute the $(n-1)^{\text{th}}$ GDH partial key, $(n-1)$ to send to M_n items, and $(n-1)$ to send combinations to each M_i. Note that the overhead of the $(n-1)^{\text{th}}$ GDH partial key computation steps has been decreased, since partial factors are not transmitted during these steps.

The same authors proposed another simple hybrid solution by adding the same principle (password authentication addition) to the hyper-cube based key agreement. The idea is merely to use the basic authenticated two-party DH in

[7] *Note that this approach has been proposed for group key, but it can be generalized to general private key.*

[8] *See the previous description of the two-party DH for more details.*

[9] *More details about this combination computation and the protocol steps can be found in [58].*

each step. Consequently, the hyper-cube based key agreement computation complexity is multiplied by 2 in this hybrid solution.

Another hybrid solution was proposed in [60], in which Li *et al.* suggest the combination of centralized and key agreement approaches. The main idea is to cluster nodes in groups according to their physical position, and to associate a leader (connector) to each group. The key agreement protocol is executed between these connectors, which distribute the resultant key to members of their groups. This solution is general and lacks detailed specifics about which key agreement protocol to use, which is indeed problematic. Moreover, the groups should be reconstructed when nodes change their position, which requires important overhead and extra computation steps when mobility increases.

Table 4 summarizes some features of the n-party solutions discussed in this section with respect to different issues. The first issue is distribution. Some solutions are totally distributed, and all nodes participate in the execution of the protocol; others are partially distributed since the protocol execution is restricted to a subset of nodes. GDH, hypercube-based, octopus, and cluster-based solutions do not require any pre-sharing of secrets; the others require a pre-sharing of some password. Except for the standard n-party password-based approach, all the other solutions are contributory, i.e., the key is not chosen unilaterally by one node but different nodes contribute to its selection. The computation complexity of each solution discussed in this section is given in the table. Note that the computational complexity of the cluster-based hybrid protocol depends on which key agreement protocol is used, which was not specified.

Rekeying

An important issue related to private key, especially in *dynamic* group communication (relying on symmetric group key), is *rekeying*. The group key must be revoked and redistributed to all the remaining nodes in a secure, reliable, and timely fashion whenever group membership changes, particularly when a node leaves the network. This problem has been extensively studied in the context of wired networks and several protocols have been proposed.

Rekeying schemes have been categorized into two classes: *stateful* and *stateless* [61]. The stateful class includes several protocols based on the use of logical key trees, such as LKH [62] and ELK [63]. In these protocols, keys are encrypted by the key server before distribution. The key server uses key encryption keys (keys used to encrypt keys) that were transmitted to members during previous rekeying operations to encrypt the keys to be transmitted in the current rekeying operation. Thus, a member must receive all the key encryption keys of interest in all the previous rekeying operations, otherwise it will not be able to decrypt the new key.

Stateless group rekeying protocols [64–66] form the second class of rekeying protocols. In these protocols, a legitimate user only needs to receive the key of interest in the current rekeying operation to decode the current group key. The stateless feature makes these protocols very attractive for dynamic ad hoc networks. However, all these protocols have much higher communication overhead than the stateful protocols. The protocols of both classes are centralized and not directly applicable to ad hoc networks.

Thus far, only a few works have been devoted to rekeying in ad hoc networks. Lazos and Poovendran [67] have proposed a new solution by adapting the LKH scheme for ad hoc networks, but the proposed solution suffers from high communication and computation overhead [61]. Another novel solution devoted to MANETs has been proposed by Kaya *et al.* [68]. This solution uses public-key techniques and is somewhat expensive in both communication and computation. An interesting protocol with regard to overhead is GKTVIPAN [61]. However, this protocol provides only *partial statelessness*. Furthermore, the simulation study performed to asses GKTVIPAN [61] has solely considered a static network, thus the satisfactory results only reflect a network without node mobility. We think mobility is a very important feature of MANETs, so an efficient re-keying scheme should consider this issue, and should require low cost for distributing the new key in a mobile environment. Rekeying remains an open research topic, and proposing a fully stateless and efficient protocol that takes into account MANETs' features, especially the infrastructureless and nodes' mobility, is a challenging problem.

Group Communication and Private Keys

Group communication technologies have proven their importance in different fields of daily life, such as education, entertainment, and other industries. E-learning and video conferences are examples of applications that belong to these fields.

Group communication, both one-to-many or many-to-many, has become increasingly important in ad hoc networks. In MANET group communications, issues differ from those in traditional networks because of the variable and unpredictable nature of the wireless medium, resource limitation, the infrastructureless nature, and the nodes' mobility.

Multicast routing in ad hoc networks can be classified into tree-based and mesh-based approaches [69]. In tree-based multicast routing protocols data packets are forwarded on a single path to a given receiver. The union of the paths to all receivers forms the multicast-tree, which may be sender specific or common to all senders in a multicast session. Examples of tree-based multicast routing protocols for MANETs are: Bandwidth Efficient Multicast Routing Protocol [70], Multicast Ad Hoc on Demand Distance Vector Routing (MAODV) [71], Multicast Core Extraction Distributed Ad Hoc Routing (MCEDAR) [72], and Adaptive Demand-Driven Multicast Routing (ADMR) [73]. Typically these approaches include mechanisms such as local repair of the distribution tree or backup paths in order to compensate for the frequent topological changes in an ad hoc network. In mesh-based approaches there may be multiple paths to each receiver. This redundancy provides increased protection against topological changes. Examples of mesh-based multicast routing protocols for mobile ad hoc networks are On Demand Multicast Routing Protocol (ODIVIRP) [74], Core-Assisted Mesh Protocol (CAMP) [75], Neighbor Supporting Ad hoc Multicast Routing Protocol (NSMRP) [76], and Dynamic Core Based Multicast Routing Protocol (DCIVIP) [77].

A private group key infrastructure is essential to secure such multicast communication, regarding both routing and data information. Moreover, in this type of communication the group membership is dynamic, i.e., nodes leave and join the group continuously. Therefore, an efficient re-keying mechanism is mandatory to ensure a robust multicast system.

Providing efficient group communication is one of the main issues in MANETs. The nature and the features of ad hoc networks make this issue even more challenging. While these networks are rapidly gaining popularity, there is a strong need to develop efficient strategies to support group communication. In particular, it is essential to suggest new efficient solutions for private key distribution and revocation that cope with MANET features.

PUBLIC KEY INFRASTRUCTURE

Thus far, we have presented solutions related to private key management. We will now discuss the public key management problem.

In a public key infrastructure, each node has a public/private key pair. Public keys can be distributed to other nodes, while private keys should be kept confidential to individual nodes. This type of key is essential for any service or application that employs asymmetric cryptography, such as many of the protocols described earlier, which use digital signature. In a traditional public key infrastructure, there is a trusted entity called a certification authority (CA) that distributes nodes' public keys in *certificates*. The CA has a public/private key pair. The private key is used to sign certificates binding public keys the CA provides for nodes, while the public key is used by nodes to check the certificate's authentication.

However, it is problematic in MANETs to establish a key management service using a *single* CA. A standard approach to improve service availability is replication, but a naive replication of the CA makes the service more vulnerable, since compromising any single replica that possesses the service private key could lead to the collapse of the entire system. To solve this problem, recent solutions propose to distribute the trust over a set of nodes by letting theme share the key management responsibility. In the following sections we present these solutions.

First General Solution

Zhou and Haas [78] are the first to have addressed public key management in MANETs. They have proposed the use of threshold cryptography [35] in order to distribute the trust over a set of nodes. In their approach, the key management service with a $(n, k + 1)$ configuration $(n \geq 3k + 1)$ consists of n special nodes, called servers. Each server stores public keys of all the nodes in the network, and knows the public keys of other servers. Thus, servers can establish secure links among them. The authors assume that the adversary cannot compromise more than k servers in any period of time with a certain duration.

The proposed $(n, k+1)$ threshold cryptography scheme allows n parties to share the ability to perform a cryptographic operation (e.g., creating a digital signature), so that any $k + 1$ parties can perform this operation jointly, whereas it is not feasible for at most k parties to do so. Therefore, the service private key k is divided into n shares, assigning one share to each server. For the service to sign a certificate, each server generates a partial signature for the certificate using its private key share and submits the partial signature to a *combiner* (any server). With $k + 1$ correct partial signatures, the combiner will be able to compute the signature for the certificate. To ensure that a compromised combiner cannot prevent a signature from being computed, the authors suggest to use $k + 1$ servers as combiners. This ensures that at least one combiner is correct and is able to compute the signature, since the maximum number of compromised severs is assumed not to exceed k. A combiner can verify the validity of a computed signature using the service public key. In case a verification fails, the combiner tries another set of $k + 1$ partial signatures. This process continues until the combiner constructs the correct signature.

To make the solution fault tolerant (especially to tolerate mobile adversaries [79]), share refreshing has been proposed [78]. This technique consists of rebuilding new sharing of the same service private key SK, and is performed as follows. First, each server randomly generates $(s_n, s, \ldots, s_{in})$, an $(n, k+1)$ sharing of 0. Then every sub-share s_{ij} is distributed to server j through a secure link. When server j gets the sub-shares $s_{1j}, s_{2j}, \ldots s_{nj}$, it just sums and adds them to its old shares s_j to get its new share s'_j.[10]

This first solution is very general, and it lacks precision about how to select servers. An important drawback of this solution we note is that the communication pattern of the proposed protocol between a collector and $k + 1$ or more servers is one-to-many-to-one (manycast), viz. A collector asking for an authentic certificate of some node needs to contact at least $k + 1$ servers and to receive at least $k + 1$ replies, which requires important overhead and long delay. Another drawback is related to the parameter k. The authors suggest that this parameter should reflect the maximum number of nodes that can be simultaneously compromised by a malicious node, but no specifics as to how to determine k's value have been illustrated. On the other hand, this solution tackles the problem of fault tolerance, especially against mobile adversaries [79], and proposes the efficient technique of share refreshing.

MOCA

MOCA [80] is another solution that employs threshold cryptography to distribute CA functionality over specially *selected* nodes. The choice of these nodes or MOCAs (MObile Certificate Authorities) is based on their security and their physical characteristics. That is, mobile nodes may be heterogeneous in many respects, especially in terms of their security. In this case, the most secure nodes would be selected as MOCAs. To illustrate this heterogeneity, the authors gave the example of a battlefield scenario, in which nodes are soldiers that have different ranks. According to their ranks, nodes would be equipped with computers that are different in power, capabilities, transmission ranges, levels of physical security, and so on. In such a case, the most secure and powerful nodes (of the most ranked soldiers) should be selected as MOCAs. In the proposed protocol (MOCA certification Protocol, or MP), a client that requires a certification service sends certification request (CREQ) packets, then any MOCA that receives a CREQ accordingly responds with a certification reply (CREP) packet containing its *partial signature*. The client waits a fixed period of time for k such CREPs. When the client collects k valid CREPs, it can construct the *full required signature* and the certification request succeeds. Hence, unlike the previous solution, in this one the requestor itself acts as a combiner. If too few CREPs are received, the client's CREQ timer expires and the certification request fails. On failure, the client can retry or proceed without the certification service. The CREQ and CREP messages are similar to route request (RREQ) and route reply (RREP) messages of the on-demand routing protocols discussed earlier.

The shape of a MOCA framework is determined by the total number of nodes in the network, the number of MOCAs, and the threshold value for secret reconstruction (number k of MOCAs required to issue certificates). Although the total number of nodes in the network (M) can change dynamically over time, it is not a tunable parameter. The number of MOCAs (n) is determined by the characteristics of nodes in the network, such as physical security or processing capability, and it is also not tunable. Given M and n, the last parameter k, which is the threshold for secret recovery, is indeed a tunable parameter.

After k chosen and the system deployed, it is expensive to change k. Therefore, it is important to understand the effects of varying k on a given system. k can be chosen between 1 (a

10 $s'_j = s_j + \sum_{i=1}^{n} s_{ij}$ is a new share of : $SK + \sum_{i=1}^{n} 0 = SK$

Solutions	Distribution	Based on threshold crypto	Collector when using threshold crypto	Overhead	Latency	Guarantee
First solution	Partially	Yes	Any server	Important	Important	Deterministic if no partition
MOCA	Partially	Yes	The requestor	Important	Important	Deterministic if no partition
Certificate chain based	Fully	No		Moderate	Short	Probabilistics

Table 5. *Public key management solutions.*

single MOCA) and n (all MOCAs). Setting k to a high value has the effect of making the system more secure against possible adversaries since k is the number of MOCAs an adversary needs to compromise to collapse the system. But at the same time, a high value can cause huge communication overhead for clients since any client needs to contact at least k MOCAs to get a certificate. Therefore, the threshold k should be chosen to balance the two conflicting requirements, and it is clear that no one value will fit all systems.

It is possible that an ad hoc network does not have enough heterogeneity among its nodes, which may make it difficult or even impossible to choose MOCAs based on this heterogeneity assumption. In such cases the authors suggest choosing *randomly* a subset of nodes as MOCAs. We think this selection might be inefficient, hence a rigorous selection strategy that fulfils a defined set of criteria, such as decreasing the overhead or the latency, is required. This still represents an open research area. Note that as with the previous solution, MOCA also suffers from the high overhead and long delay.

Certificate Chain-based Solution

Capkun *et al.* [81, 82] have proposed a fully distributed self-organizing public key management system in which nodes generate their keys, and then issue, store, and distribute public-key certificates. This system is similar to PGP (Pretty Good Privacy) [83] in the sense that the public-key certificate of each node is issued by the node itself. However, in order to remove the reliance on on-line servers (which are clearly incompatible with MANETs), the proposed system does not rely on certificate directories [83] for the distribution of certificates. Instead, certificates are stored and distributed by the nodes, and each node maintains a local certificate *repository* that contains a limited number of certificates selected by the node according to an appropriate algorithm.

When node u wants to verify the *authenticity* of v's public key, the two nodes (u and v) merge their local certificate repositories, then u tries to find an appropriate certificate chain from u and v in the merged repository. An algorithm for the construction of local certificate repositories has been proposed [81]. The basic operations of this public-key management scheme are as follows.

Creation of public keys: The public key and the corresponding private key of each node is created locally by the node itself. Note that no details about key creation have been fixed in the two previous solutions.

Issuing public-key certificates: If a node u believes that a given public key K_v belongs to a given node v, then u can issue (using its own signature) a public-key certificate regarding K_v. There may be many reasons for u to believe that K_v belongs to v, for instance, u may receive K_v on a secure channel that is associated with v, or someone trusted by u claims that K_v belongs to v.

Storage of certificates: Certificates issued in the system are stored by nodes in a fully decentralized way. Every node maintains a local certificate repository that has two parts. First, each node stores the certificates that it issued. Second, each node stores a set of additional certificates issued by other nodes, which are selected according to an appropriate algorithm given in [81]. This additional set of certificates is obtained from other nodes, thus some underlying routing mechanisms are assumed to exist.

Key authentication: When a node u wants to obtain the authentic public key K_v of another node v, it asks other nodes (possibly v itself) for K_v. In order to verify the authenticity of the received key, the requested node provides u with its local certificate repository, then u merges the received repository with its own repository and tries to find an appropriate *certificate chain* from K_u to K_v in the merged repository. If this fails, u may ask other nodes for further certificates.

Unlike the two previous solutions, this approach is fully distributed and more suitable for MANET environments, and it also provides more precision as to the issuing of public keys. Moreover, this solution does not use threshold cryptography, and it decreases both the overhead and the latency, since a requestor needs only to ask one node for the certificate instead of asking k different servers.

However, we should note that this approach provides only *probabilistic* guarantees, viz. it does not ensure to a requestor node that it will get a certificate, since no node saves keys of all the other nodes. On the other hand, in the previous solutions collectors maintain keys of all nodes, so these solutions are deterministic and guarantee that a correct certificate will be provided if no adversary can simultaneously compromise enough servers to jeopardize the solution,[11] provided that there is no partition in the network preventing nodes to contact the servers. All these features are listed in Table 5.

Intrusion Detection Systems (IDSs)

An intrusion may be defined as "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource" [84], or "any unauthorized or unwanted activity on a system or a network" [85]. An IDS may be defined as "a system that tries to detect and alert on attempted intrusions into a system or a network" [85].

The history of security research has taught us a valuable lesson: no matter how many intrusion prevention measures are inserted in a network, there are always some weaknesses in the systems that one could exploit to break in [86]. These weaknesses include design and programming errors, and various social engineering penetration techniques as well. Hence, intrusion prevention measures (proactive solutions) cannot eliminate attacks, and they must be fortified with IDSs. An IDS presents a second wall of defense and is essential for any high-survivability network.

The primary assumptions of intrusion detection are:

[11] *$k + 1$ servers for the first solution and K for the second.*

- User and program activities are observable, for example via system auditing mechanisms.
- More importantly, normal and intrusion activities have distinct behavior.

Therefore, intrusion detection involves capturing *audit data* and reasoning about the evidence in the data to determine whether the system is under attack. There are mainly two classes of IDSs:

Anomaly detection: These IDSs consider activities that deviate significantly from the established normal usage profiles as anomalies, i.e., possible intrusions, where "normal" patterns are defined beforehand. For instance, the normal profile of a user may contain the averaged frequencies of some system commands used in his login sessions. If, for a session that is being monitored, the frequencies are significantly lower or higher, then an anomaly alarm will be raised. The main advantage of anomaly detection is that it does not require prior knowledge of intrusions and can thus detect new intrusions. The main disadvantage is that it might be unable to describe what the attack is and might have a high false positive rate. An exemple of this type of IDS in traditional networks is IDES [87]

Misuse detection (signature-based): These IDSs rely on the use of specifically known patterns of well known unauthorized behavior and attacks to match and identify *known* intrusions. For instance, a rule for the "guessing password attack" can be "there are more than four failed login attempts within two minutes" [86]. The main advantage of this technique is that it can accurately and efficiently detect instances of known attacks. Its main drawback is that it lacks the ability to detect the truly innovative (i.e., newly invented) attacks, whose patterns are unknown. IDIOT [88] and STAT [89] are examples of signature-based IDSs in traditional networks.

Another classification of traditional IDSs is based on the type of audit data used. This class includes:

Network-based IDS: Normally, an IDS of this category runs at the gateway of a network, where it captures and examines packets that go through the network hardware interface.

Host-based IDS: Relies on operating system audit data to monitor and analyze the events generated by programs or users on a host.

An intrusion detection model has two components: the *features* (attributes or measures) and the *modeling algorithm*. Defining a set of predictive features that accurately capture the representative behaviors of intrusive or normal activities is the most important step in building an effective intrusion detection model, and it can be independent of the design of the modeling algorithm. This latter is a rule-based pattern matching that uses the features to identify intrusions.

Problems of Traditional IDSs

The vast difference between traditional networks and MANETs makes it very difficult to directly apply traditional intrusion detection techniques to MANETs. The most important difference is the infrastructureless nature of MANETs. In other words, unlike wired networks where traffic monitoring is usually done at switches, routers, and gateways, *a MANET does not have such traffic concentration points* where the IDS can collect audit data for the entire network. The second big difference consists of a MANET's limitations, presented earlier. IDS modules, when implemented within MANET nodes, should take these limitations into account and should not consume too many resources (battery, CPU, bandwidth, etc.).

Another big problem with MANETs is the lack of clear separation between normalcy and anomaly. For instance, a node that sends out false routing information could be merely someone that has temporarily out-of-date routing information, due to volatile physical movement. Intrusion detection may find it increasingly difficult to distinguish false alarms from real intrusions.

Reachable	Delivered	Cached
True	True	True
True	False	False
False	False	True
False	False	False

Table 6. *Normal events.*

Anjum *et al.* [90] have investigated by simulation the ability of various MANET routing protocols to facilitate intrusion detection to a traditional host signature-based IDS. More precisely, they have measured the detection rate of some attacks whose signatures are assumed to be known by the IDS's modules, these latter having been assumed to perform *independently*. The results show that the detection rates are dramatic, especially when the nodes' mobility increases. This illustrates the inefficiency of traditional host-based IDSs when directly applied to MANETs. We also realize that node cooperation is mandatory.

Novel Solutions

Recently some IDSs have been proposed for MANETs; they are all distributed, host-based, anomaly-based, and cooperative. The cooperation, however, may be fully and equally distributed among nodes, or it may be based on hierarchal node organization. The IDS design differs from one solution to another, as we will describe in the next section.

Correlation-based IDS

In [91] a correlation and learning-based approach for constructing anomaly detection models for MANET routing protocols has been proposed. The authors claimed that strong feature correlation exists in normal behavior, and that such correlation can be used to detect deviations caused by abnormal (or intrusive) activities. To explore such a correlation, the authors have developed a cross-feature analysis anomaly detection approach, and propose to compute a *classifier* to each defined feature. Each classifier is *learned* from a set of *training data*, and helps compute the most likely value of the corresponding feature given the values of other features (conditional probability).

The original anomaly detection problem, i.e., whether a record (a set of observed values of the features) is normal or not, is solved as follows. Given a record, first apply each classifier to compute the probability of the appropriate feature, given the values of the others, then compute the average probability and compare it with a *predefined* threshold. If the average probability is lower than the threshold, an alarm is raised. To clarify this approach, we give the following simple illustrative example [91].

Example: Consider a simple ad hoc network with two nodes. Packets can only be sent from one end to the other if they are within each other's transmission range. The following three features are defined:

- Is the other node *reachable*?
- Is there any packet *delivered* during the last five seconds?
- Is there any packet *cached* for delivery during the last five seconds?

For simplicity, all features are assumed to take binary val-

Reachable	Delivered	Cached	Class	p(r/d,c)	p(d/r,c)	p(c/r,d)	Average
True	True	True	Normal	1	1	1	1
True	True	False	Abnormal	0.5	0	0	0.17
True	False	True	Abnormal	0	0	0	0
True	False	False	Normal	0.5	1	1	0.83
False	True	True	Abnormal	0	0	0.5	0.17
False	True	False	Abnormal	0.5	0	0.5	0.33
False	False	True	Normal	1	1	0.5	0.83
False	False	False	Normal	0.5	1	0.5	0.67

Table 7. *Probabilities table.*

ues, i.e., either true or false. All normal events are enumerated in Table 6, and all the other combinations not enumerated in this table are abnormal. For instance, *delivered = true* while *reachable = false* is an abnormal situation since no delivery is possible if the corresponding node is unreachable. Therefore, (*false*, *true*, *true*) and (*false*, *true*, *false*) are abnormal events (records).

An illustrative classifier can be used in this example that works for each feature f_i as follows:

- If only one record is normal where the two other features have been assigned with a particular set of values, the record is selected as the predicted one with the associated probability of 1, i.e., probability of v_i (the possible value of f_i) given the other values is 1. For instance, when a packet is delivered and cached during the past five seconds (*delivered = cached = true*), the corresponding node is inevitably reachable, thus the probability of *reachable = true* given *delivered = cached = true* is 1.
- If both records are normal, the associated probability of each one is 0.5. For example, when no packet is neither delivered nor cached during the past five seconds (*delivered = cached = false*), the corresponding node could be either reachable or unreachable. The probability of *reachable = true* (respectively, *reachable = false*) given *delivered = cached = false* is thus 0.5.
- If both are abnormal, the associated probability of each one is 0.5.

Table 7 shows conditional probabilities and their averages for each possible record; the features reachable, delivered, and cached are denoted, respectively, by r, d, and c. As can be seen in Tables 6 and 7, a threshold of 0.5 allows the correct classification of all the events.

In the proposed IDS, this table is not pre-computed and saved by nodes, but each line is computed when the appropriate record is observed during the monitoring period (five seconds in this example). For instance, when the record (false, true, true) is observed, $p(r = false/d = true, c = true)$, $p(d = true/r = false, c = true)$ and $p(c = true/r = false, d = true)$ will be computed (using the classifier), whose values are, respectively, 0, 0, and 0.5 . These probabilities will be averaged and compared with the threshold, then an alarm is raised since the average value (0.17) is bellow the threshold.

This simple example with two nodes has been given for illustration only. In their study, however, the authors defined a total of 141 more complicated features, which capture the basic view of network topology and routing operations, as well as traffic patterns. During their simulation the authors used trace data of normal runs for training the anomaly detection models; the set of classifiers and the threshold are computed in this phase. Afterward, they ran a set of various attacks such as, packet dropping, routing loop creation, sleep deprivation etc., on the AODV routing protocol (presented earlier), and collected the trace data for evaluating the model. First, an anomaly detection model computes features and detects anomalies locally on each node. As long as there is one detector (on one node) that identifies an abnormal event, this latter is counted as an anomaly alert (true detection or false alarm) for the network. The results show that the model has good performance in detecting anomalies. To provide more accurate information about the attack, its type, and the attacker, whenever possible, cooperation among nodes is mandatory.

In their subsequent work [85] the authors proposed a cooperative IDS based on the previous anomaly detection model, where they have defined rules to be activated whenever some anomaly has been observed. The rules have been defined for a given set of attacks and are based on traffic analysis in promiscuous mode. When an alert is provided by the anomaly detection model, the monitoring module (either of the same node or of another node, according to the defined rules related to the anomaly) is activated to determine the attack. The simulation results show satisfactory results in detecting the type of attack, and in the false alarm rate, especially regarding packet dropping and sleep deprivation (using flooding) attacks presented in the previous sections.

The satisfactory simulation results reflect the application of the model computed from a sample network to the same one. We think the feature correlation may change from an application to another, and from a network to another. In this case, the learning phase is required each time at the initialization of the network. This requirement might be inappropriate for dynamic ad hoc networks. Therefore, further investigation into different configurations is required, and can represent a research topic.

Cluster-based IDSs

In the previous cooperative solution, each node participates in the monitoring to detect the type of attacks. However, when the threat level is low, it might be inefficient to keep each MANET node always monitoring, since MANET nodes typically have limited battery power. Instead, Hung and Lee [85] have suggested that *periodically* each cluster of neighboring nodes *randomly* and *fairly* elect a clusterhead to lead IDS functions for the entire neighborhood (cluster). It instructs the other members on how the feature computation is to take place.

For this purpose, the authors have proposed a clusterhead

assignment algorithm that aims at fairly electing a clusterhead, in such a way that every node is a member of at least one cluster. A cluster is defined as a group of nodes that are in the same neighborhood, that is, each node of the cluster is within a one-hop vicinity of each other. As a special case, a node that cannot be reached by anyone else forms a single node cluster. The proposed algorithm ensures randomness in election decisions, and also implements *periodical* re-election to guarantee equal service time for every node. The algorithm also guarantees that no node can manipulate the selection process to increase (or decrease) the chance for it (or another node) to be selected. A detailed description of this algorithm can be found in [85]. After clustering, two detection schemes that define how and where features are computed and transmitted have been proposed:

- LFSS (Local Feature Set Scheme): In this scheme, feature computation is still done locally on each node, but anomaly and attack detections are performed by the clusterhead. For each feature sampling period, a randomly selected cluster member (which can be the clusterhead itself) is requested to transmit its whole feature set to the clusterhead.
- CLFSS (Clusterhead-Assisted Local Feature Set Scheme): In order to reduce the burden on cluster members and the traffic overhead, the clusterhead can help compute some of the features, more specifically traffic-related features. In this scheme the clusterhead overhears incoming and outgoing traffic on all members of the cluster. Corresponding traffic-related features are computed in the same way as a local node does in LFSS, as if the cluster is a large node as a whole. Nevertheless, the members (more precisely, one member at a time) are still responsible for computing and transmitting other features (non traffic-related features) to the clusterhead, as in LFSS.

The simulation results show that CLFSS outperforms LFSS. We think this approach (cluster-based) might cause significant overhead for clusterhead reconstruction when node mobility increases, since clusters are totally depending on the network topology. Moreover, no comparison between the previous approach and this approach has been made. Does the cluster-based approach really provide improvement? This remains an open question.

Agent-based IDSs

Cooperative Agent-based IDS — Zhang *et al.* [86] have proposed a novel agent-based architecture in which every node equally participates in intrusion detection and response. Each node is responsible for detecting signs of intrusion (anomalies) locally and independently, but neighboring nodes can collaboratively investigate in a broader range.

In the systems aspect, individual IDS agents are placed on each node. Each IDS agent runs independently and monitors local activities, including user and system activities, as well as communication activities within the radio range. It detects intrusion from local traces and initiates response. If an anomaly is detected in the local data but the evidence is inconclusive, then neighboring IDS agents will cooperatively participate in global intrusion detection actions. These individual IDS agents collectively form the IDS system to defend the MANET.

The internal of an IDS agent can be conceptually structured into six pieces:

- Data collection module: Responsible for gathering audit data from various sources, including system and user activities within the mobile node, communication activities of this node, and the observable communication activities within the node's neighborhood. Therefore, multiple data collection modules can coexist in one IDS agent.
- Local detection engine: Uses data collected by the data collection module to detect local anomaly.
- Cooperative detection engine: Used by the detection methods that need broader data sets or that require collaborations among IDS agents.
- Local response module: Triggers local actions in the mobile node.
- Global response module: Coordinates actions among neighboring nodes.
- Secure communication module: Provides a high-confidence communication channel among IDS agents.

This framework is very general and can be used to design an IDS that includes different layers, provided that the anomaly detection model defines features and captures attacks on these layers. For example, at the MAC level the following features could be considered: the total number of channel requests during a given period *s*, the number of nodes making a request, etc. At the application layer, the features could include: the total number of requests to the same service, the number of different services requested, the average duration of a service, etc. The authors extended their work by dealing with the network layer, and they proposed an anomaly detection approach based on classifiers and feature correlation (as the first presented solution), which also requires a training phase and has the same drawbacks.

Clustered Agent-based IDS — Kachirski and Guha [92] have proposed an agent-based IDS, consisting of three types of agents: monitoring, decision making, and action. Monitoring agents are responsible for both network (packet-level) and host (user-level and system-level) monitoring.[12] Decision agents make decisions about the intrusion detection evidence, while action agents take action when an intrusion is detected by the intrusion detection agents. A clustering algorithm has been proposed that elects clusterheads in charge of decision making and *network* monitoring agents; note that the other types of agents are present in every node. An important input parameter for this algorithm is the maximum number of hops a clusterhead is located from any other member. Fixing this parameter to one means that each node will have at least one neighboring node hosting a network monitoring agent (clusterhead), which is idem to the clustering algorithm of the second solution. This will provide more accuracy but requires significant overhead. To preserve node resources, a two-hop cluster scheme has been proposed in which the maximum number of hops between any node and the clusterhead is two. For the same purpose, the size of queues used for packet monitoring has been *limited*, which allows IDS overhead to be limited. Unlike the previous agent-based solution, decision making in this solution is not totally cooperative but *independent*. That is, not all nodes are responsible for making decision about IDS evidence, but only those hosting decision making agents (clusterheads). These agents use information provided from the local monitoring agents as well as those regarding network traffic provided by the network monitoring agents, and they collaborate to make decisions. Upon detection of an intrusion, the action agent hosted by the intruder node will be informed.

The anomaly detection model has not been specified and is left for future work. The author's perspectives also include a

[12] *We can thus distinguish two subcategories: network monitoring agents and host monitoring agents.*

Solutions	Cooperation	Correlation-based	Cluster-based	Agent-based	Drawbacks
Correlation-based	Totally	Yes	No	No	All nodes have to monitor network traffic Requires a learning phase
Cluster-based	Partially	Yes	Yes	No	Overhead for clusters reconstruction Requires a learning phase
Cooperative agent-based	Totally	Yes	No	Yes	All nodes have to monitor network traffic Requires a learning phase
Clustered agent-based	Partially	No	Yes	Yes	Overhead for clusters reconstruction, but less than the second solution It is very general, and lacks specifications about the anomaly detection model

Table 8. *Main features of the presented IDSs.*

novel cooperative detection algorithm, as well as investigations of possible attacks on the proposed IDS.

Table 8 summarizes the main features and drawbacks of the solutions presented in this section. Note that *correlation-based IDS* is the *cooperative* approach [85] presented earlier, and not the basic approach. Both the first and the third solutions are *totally* cooperative, i.e., all nodes equally participate in monitoring and intrusion decision. On the other hand, cluster-based IDSs are partially cooperative, viz not all nodes participate in network monitoring and intrusion decision; only clusterheads are responsible for these tasks. These solutions have overhead for cluster reconstruction, especially when nodes' mobility increases. The last solution decreases this overhead by employing the two-hop cluster scheme. In the totally cooperative solutions every node has to monitor all the traffic in its vicinity, which might be inefficient when the threat level is low. Excepting the fourth solution, which is very general and lacks specifics about the anomaly detection model to use, all the other solutions are correlation-based, and thus might require a learning phase, which is inappropriate for dynamic MANETs.

Security Issues in Wireless Sensor Networks (WSN)

Wireless sensor networks, applied to monitoring physical environments, have recently emerged as an important application of the ad hoc network paradigm. This technology has mainly been made possible by the convergence of micro-electro-mechanical systems technology, wireless communications, and digital electronics, enabling the construction of low-cost, low-power, multifunctional sensor nodes that are small in size and communicate untethered in short distances, thus forming the sensor network [93].

A sensor network consists of thousands, even millions of tiny devices equipped with signal processing circuits, microcontrollers, and wireless transmitters/receivers, in addition to embedded sensors. Nodes are randomly and densely deployed over the sensing field, leading therefore to a need for auto-organization capability. Potential applications of sensor networks include, but are not limited to, geophysical monitoring (seismic activity), precision agriculture (soil management), habitat monitoring (tracking of animal herds), target tracking in battlefields, and disaster relief networks [94].

Early research efforts have focused on the development of a new network protocol stack, trying to meet performance requirements that are more stringent than in other ad hoc networks, including energy efficiency, auto-organization, scalability to a high number of nodes, etc. However, most applications of sensor networks face acute security concerns, including eavesdropping, forgery of sensor data, denial of service attacks, and the physical compromise of sensor nodes [95]. This renders security as important as other performance issues so that we can make such technology viable. In the following section we assess security threats specific to sensor networks and then describe the appropriate security mechanisms that were adapted or newly created to meet distinctive features of this special class of ad hoc networks (constrained energy consumption, limited computational resources, etc.). We consider two classes of attackers: mote-class attackers and laptop-class attackers [96]. In the former case, the attacker is of the same nature as the sensor nodes, so it has access to some few sensors with the same capability. In contrast, laptop-class attackers may have access to more powerful devices such as laptops or their equivalent, so it can perform more powerful attacks such as jamming or eavesdropping the entire sensor network using its stronger reception power.

Communication Security Constraints in WSN

Sensor nodes for large-scale WSN have very limited capabilities in term of available energy, memory capacity, and processing speed. They vary from those called smart dust sensors [97] that have only 8 KB of program and 512 bytes of data memory, and processors with 32 8-bit general registers that run at 4 MHZ and 3.0 V (e.g., ATMEL 90L58535), to sensors such as mica2dot [98] that are over an order of magnitude more capable in term of processing speed (e.g., Atmel ATmega28L.), and memory size (128 kb of program flash memory). The available range of capability makes it impractical to use typical asymmetric (public-key) crypto systems to secure communication. Caraman *et al.* [106] pointed out that on a mid-range processor such as the Motorola MC68328 "DragonBall," the energy consumption for a 1024-bit RSA encryption (respectively signature) operation is much higher than that for a 1024-bit AES encryption operation; i.e., about 42 mJ (respectively 840 mJ) versus 0.104 mJ. Furthermore, the energy consumption for transmitting a 1024-bit block over a distance of approximately 900 meters using a typical communication subsystem such as Sensoria WINS NG RE at 10 kb/s and 10 mW of power is about half that of RSA encryption (i.e., 21.5 mJ), and even less for reception (14.3 mJ) [100]. It has been reported also that symmetric-key ciphers and hash functions are between two and four orders of magnitude faster than digital signatures. All theses factors make symmetric-key ciphers, low-energy encryption modes, and

hash functions the systematic tools of choice for protecting WSNs.

We note here that little research work has been conducted to investigate the development of security analysis models for ad hoc and sensor networks, especially those used for the quantitative performance evaluation of encryption algorithms, in terms of communication overhead and computational cost. In fact, Ganesan *et al.* show in [101] that, based on experimental tests, an analytical model can be derived to assess the impact of arbitrary architectures on different encryption schemes. Their model consists of a multi-variant function that is dependent on several parameters such as architectural features, namely processor frequency, ISA type, etc. We believe that such models [101–103] allow designers to project computational limitations and determine the threshold of feasible encryption schemes under a set of constraints for a given embedded architecture such as sensor nodes.

Key Management Issues

Problems

Although "key management" is important for ensuring confidentiality and authentication, it still remains an unsolved problem in WSNs, mainly due to the following problems.

Key Pre-Deployment: Because of the unknown physical topology prior to installation of the sensor network, predeployment keying is considered as the only practical option that the key distribution phase would have to rely on [100]. However, traditional key pre-deployment schemes are inadequate for WSNs, where the installed keys on each node are either a single mission key or a set of separate $n - 1$ keys, each being pair-wise privately shared with another node. In fact, in the former case the capture of any sensor node may compromise the entire network, because selective key revocation is impossible upon sensor capture detection, whereas the pair-wise key sharing solution requires storage and loading of $n - 1$ keys on each sensor node. This becomes impractical when using more than 10000 sensors, due to the resources limitations described previously. Moreover, pair-wise private key sharing between any two sensor nodes would be unusable since direct node-to-node communication is achievable only in small node neighborhoods.

Shared Key Discovery: Another challenging issue is that each node needs to discover a neighbor in wireless communication range with which it shares at least one key. Thus, a link exists between two sensor nodes only if they share a secret key. A good shared key discovery approach should not permit an attacker to know shared keys between every two nodes.

Path-key Establishment: For any pair of nodes that do not share a key and are connected by multiple hops, there needs to be assigned a path-key to guarantee end-to-end secure communication. It is important that the path-key should be different from pair-wise shared keys of intermediary nodes.

In addition to these problems, key management in sensor networks faces other important challenging issues such as building energy-efficient re-keying mechanisms when available keys are expired, as well as minimizing key establishment delays.

Solutions

To overcome the shortcomings of traditional predeployed keying approaches, essentially the big size of the loaded key ring on each node, several alternatives have been proposed in the literature.

For instance, the probabilistic key sharing protocol described in [100] uses a shared key discovery approach that guarantees that every two nodes can, with a chosen probability, share one key while ensuring that only a small number of keys need to be loaded on each sensor node's key ring. This latter is drawn out randomly from a large pool of keys initially generated. However, this protocol suffers from several flaws because of its reliance on trusted controller nodes where key identifiers of each key ring and the corresponding sensor's IDs are saved. Moreover, the resilience of the basic probabilistic key sharing approach to node capture is weak, since only one key is shared between every two nodes, which makes it easier for an attacker to break a greater number of secure links by capturing a small number of nodes. In an attempt to enhance the resiliency to node capture, Chan *et al.* [107] have proposed a q-composite approach where q keys ($q > 1$) need to be shared between nodes instead of just one key. When a small number of nodes is compromised, this approach has proved its efficiency compared to the basic scheme. Another important problem in the described solutions is that a shared key can be located to multiple nodes, and thus is not exclusively known by only two nodes. Hence, this key cannot be used for encrypting any message that is private to only two nodes in the network. That is why Zhu *et al.* [108] have proposed to harden the basic probabilistic approach with the use of threshold secret sharing to establish a pair wise secret key known exclusively by corresponding nodes.

So far we note that several problems remain challenging. In fact, simulation results in [109] show that the shared key discovery process in all the discussed approaches cannot resist the so called smart attacker that can selectively attack some nodes that allow it to compromise the network faster, based on already obtained information (i.e., exchanged key IDs during key discovery steps). This threat model has been considered in [109] and a new mechanism for key discovery has been proposed. This latter is based on a pseudo-random key predeployment strategy that assures a key discovery phase that requires no communications, and thus provides high resilience against the smart attacker model.

Another important research trend in the field of key management in sensor networks consists of investigating new techniques that may permit the use of public key cryptography in such environments. Indeed, Guabtz *et al.* [110] show that the right selection of employed encryption algorithms and their associated parameters, careful optimization, and low power design techniques can make asymmetric encryption feasible in sensor networks. Although this study has demonstrated that public key encryption can be achieved with an average power consumption less then 20 W using the NtruEncrypt cryptosystem [111], we note that this algorithm has not yet proved its resistance to cryptanalysis. Moreover, the considered security level of the implemented algorithms, for this study, cannot reflect a realistic scenario of using asymmetric cryptography in sensor networks, since we believe that the use of public key mechanisms in WSNs is mainly motivated by the guarantee of a higher security level than those of proposed symmetric key techniques [100, 107–109] with lower message exchange complexity.

Secure Routing

Many sensor network routing protocols are quite simple, and do not consider security as a primary goal. Consequently, these protocols are more susceptible to attacks than in general ad hoc networks. Karlof *et al.* [112] have shown how attacks against ad hoc networks and peer-to-peer networks can be adapted into powerful attacks against sensor networks. They have also introduced sinkholes and HELLO floods, two classes of novel attacks against sensor networks that we will briefly summarize.

Some New Attacks in WSNs

Sinkhole Attacks — Depending on the routing algorithm technique, a sinkhole attack tries to lure almost all the traffic toward the compromised node, creating a metaphorical sinkhole with the adversary at the center. For example, the attacker could spoof or replay an advertisement for a high quality route that passes through the compromised node. If the routing protocol employs an end-to-end acknowledgment technique to verify a route's quality, a powerful laptop-class attacker could then provide a very high-quality route by transmitting with enough power to reach the destination (sink node or base station) in a single hop, as in rushing attacks presented earlier. Since sinkhole attacks imply a great number of nodes (those on or near the high-quality route), they can enable many other attacks that need tampering with circulating traffic, such as selective forwarding. We should mention that sensor networks are particularly vulnerable to this class of attack because of their special communication paradigm, where all nodes have to send sensory data to one particular sink node. Thus, a compromised node has only to provide a single high-quality route to the sink node in order to influence a potentially large number of nodes.

Sinkhole attacks are difficult to overcome, especially in routing protocols that integrate advertised information such as remaining energy. In addition, geo-routing protocols are known as one of the routing protocol classes that are resistant to sinkhole attacks, because that topology is constructed using only localized information, and traffic is naturally routed through the physical location of the sink node, which makes it difficult to lure it elsewhere to create a sinkhole.

Hello Flood Attacks — This attack exploits Hello packets that are required in many protocols to announce nodes to their neighbors. A node receiving such packets may assume that it is in radio range of the sender. A laptop-class adversary can send this kind of packet to all sensor nodes in the network so that they believe the compromised node belongs to their neighbors. This causes a large number of nodes sending packets to this imaginary neighbor and thus into oblivion. Several routing protocols in sensor networks, such as directed diffusion [97], LEACH [99], and TEEN [104], are vulnerable to this class of attack, which may be very critical, especially when Hello packets consist of routing data or localization information exchange. One simple way to mitigate Hello flood attacks is to verify whether links are bidirectional. However, if the adversary has a highly sensitive receiver, a trusted sink node may opt for limiting the number of verified neighbors for each node in order to prevent Hello attacks.

In addition to Hello floods and sinkhole attacks, sensor routing protocols are also vulnerable to general ad hoc routing threats described earlier. In fact, these attacks can be achieved more easily and in less time then in other ad hoc networks, especially if we consider a set of laptop-class adversaries with strong transmitters and high bandwidth not available to ordinary sensor nodes, allowing such attackers to coordinate their efforts.

In an attempt to achieve major requirements for secure routing in WSNs, new protocols were proposed in the literature [96]. We describe hereafter two interesting propositions.

Solutions

SPINS — SPINS [18] presents two building block security protocols optimized for use in sensor networks, namely, SNEP and μTESLA. SNEP provides confidentiality via a chaining encryption function (i.e., DES-CBC). This technique employs a shared counter between the sender and receiver to build a one-time encryption key to prevent replay attacks and ensure data freshness. SNEP also uses a message authentication code to guarantee two-party authentication and data integrity. μTESLA, the optimized form of TESLA, is a new protocol that provides authenticated broadcasts for severely resource-constrained environments. This protocol overcomes overhead and computation problems known in asymmetric cryptographic mechanisms by introducing asymmetry through a delayed disclosure of symmetric keys [18]. Nevertheless, μTESLA requires a loose synchronization between the broadcasting node and receivers.

INSENS — INSENS [113] is a protocol that provides intrusion-tolerant routing in sensor networks by building multiple redundant paths between sensor nodes and the sink node in order to bypass intermediate malicious nodes. In addition, INSENS also limits DOS-style flooding attacks, and prevents false advertising of routing or other control information to overcome sinkhole-style attacks [114]. However, INSENS suffers from several drawbacks, the most important of which is that the sink node is supposed to be fault-tolerant, and that it cannot be isolated from the rest of the network by an attacker. This assumption is unrealistic in several scenarios.

Securing Data Aggregation

Data aggregation (or data fusion) is a key emerging theme in the design and development of WSNs. In this process, intermediary nodes called "aggregators" collect the raw sensed information form sensor nodes, process it locally, and forward only the result to the end-user. This important operation essentially reduces the amount of transmitted data on the network and thus prolongs its overall lifetime, the most critical design factor in WSNs. However, this functionality is made even more challenging due to the hostile deployment environment, which makes possible the physical compromise of aggregators and some of the sensor nodes. Indeed, possible threats can vary from denial-of-service attacks that try to stop completely this service to stealthy attacks where the attacker's purpose is to make the user accept false aggregation results. This latter is more difficult to detect. For data aggregation validity assurance Du *et al.* [115] have proposed the use of redundant data fusion nodes as witnesses. These nodes conduct the same data fusion operations as aggregators, but send the result as a Message Authentication Codes (MAC) to the aggregator itself instead of sending it to the base station. In order to prove the validity of the aggregation results, the aggregator has to forward the received proofs from witness nodes along with its calculated result to the base station. If a compromised aggregator wants to send invalid fusion data, it has to forge the proofs on the invalid results. The aggregation result is confirmed when n out of m witness proofs agree with the aggregators results, otherwise this latter is discarded and the base station polls one of the witness node to send it the valid aggregation result. We think this solution is efficient when witnesses are supposed to be trusted enough, otherwise it requires an important additional overhead to attain acceptable aggregation results using the voting scheme. Moreover, the authors have not addressed issues about choosing witness nodes. In [116] the authors have proposed a security framework based on an aggregate-commit-prove approach to verify that the answer given by aggregators is a good approximation of the true value even if the aggregators and a fraction of the sensor nodes can be corrupted. In this approach the aggregator commits to the collected data by constructing a Merkle Hash-tree [98]. The commitment ensures that the aggregator uses data provided by the sensors, and acts as a statement to be verified by the base station about the correctness of the

aggregation results. Although the authors have proposed concrete protocols for securely computing the median, average, and some other types of specific aggregation operations, we think the proposed scheme remains somewhat generic, and may not be flexible enough to support other types of in-network processing, such as in tinyDB [117]. To conclude, we believe that in-network processing is one of the key issues that has to be considered in all layers of the WSN's network protocol stack in order to minimize energy consumption. However, this operation cannot be efficiently done without being secured. For that, we think secure in-network processing should consider keying schemes that are more energy-efficient. Moreover, multi-tiered hierarchical aggregation approaches, such as in [118] would be the most efficient scheme when the WSN contains a high number of sensor nodes. For that, more research work should be undertaken on how to securely and efficiently construct such schemes and dynamically choose aggregation nodes.

Conclusion

In this article we have studied different MANET security issues, and we have shown that the special features of this new environment make it more vulnerable to threats, and that solutions developed for standard networks are often either unsuitable or not directly applicable in this environment. We dealt with several problems related to different network layers.

For the network layer we presented different types of attacks on routing protocols, and then we classified and discussed proposed techniques to mitigate these attacks. Almost all these techniques use public key encryption and thus require certificate authority (CA) for key management, which is rather problematic. We have also dealt with data forwarding, and presented eavesdropping attacks, packet dropping attacks, and selfish misbehavior, along with classifications and discussions of the proposed solutions. As shown, preventive solutions against selfish misbehavior and packet dropping only motivate nodes to cooperate or avoid losing packets. Almost all the detection solutions, on the other hand, rely on the watchdog technique, which fails to correctly detect the guilty (malicious or selfish) node in some cases, particularly when applying the power-control technique used by some power-aware routing protocols subsequently proposed in the field of power consumption optimization. A new solution that relies on a technique other than the watchdog is then required to overcome this problem. As for eavesdropping, a detection solution is required. Using encryption could help protect data confidentiality, but it is insufficient since breaking keys is always possible and key revocation in MANETs is problematic. Eavesdropping remains a serious attack against data forwarding and represents an open research topic for MANETs.

Regarding the MAC layer, which has not received enough attention in the literature, we presented the selfishness on channel access misbehavior, which breaks the fairness and greatly affects network efficiency. The only solution proposed in the literature was presented and discussed. In our discussion we illustrated how this solution may wrongly accuse well-behaving nodes, and how it is unable to detect what we called cooperative misbehavior. This problem also represents a fertile field of research.

As for the application layer, we studied the key management problem, which can also be considered as an underlying mechanism for securing lower protocols such as routing (as shown earlier). Several solutions for private key and public key management techniques were analyzed. We showed that it is a challenge to provide an efficient contributory private key solution where all nodes participate in key construction with minimum overhead and computation. In the public key infrastructure, however, we think issuing public keys is not a great problem and PGP's method (each node locally issues its own keys) can be directly adapted, as proposed in [81]. The problem in this type of key infrastructure is providing authentic and efficient key distribution. By authentic we mean that when a requestor asks for a public key of another node, the protocol should ensure that the *right* key is being provided, and that no adversary can successfully provide the requestor with a falsified one. On the other hand, efficiency means moderate overhead and latency.

Intrusion Detection Systems (IDSs), which are essential when preventive measures fail, were presented. As we saw, MANET features raise the complexity of this problem, creating a wide research area. All of the MANET IDSs we presented are host-based, anomaly-based, and fully or partially cooperative. In networks with a low threat level, it might be irrelevant to keep all nodes monitoring the traffic and equally sharing IDS tasks. Cluster-based solutions suggest the division of nodes into cluster, thereby only clusterheads will be responsible for these tasks. But the cluster construction requires overhead, especially when the nodes' mobility increases. More investigation into the efficiency of this approach (cluster-based) is required. Furthermore, the lack of clear separation between normal state and anomaly when designing the anomaly model is a great problem in MANETs. Consequently, in practice a learning phase will be required each time at the initialization of the network, which might be inappropriate for dynamic ad hoc networks. More investigation into this issue is required.

Finally, security issues related to a special type of ad hoc network, namely wireless sensor networks (WSNs), was outlined. This very constrained environment makes adaptation of existing protocols, first proposed for general ad hoc networks, a challenging endeavor. We believe the design of novel mechanisms that take into account the unique features of WSNs, such as their new communication paradigm, would be a more judicious approach.

Acknowledgments

We are grateful to Prof. Abdelmadjid Bouabdallah for his advice, support, and reception at Heudiasyc Lab of Compiegne University, where we prepared the first version of this survey. We are also grateful to our colleagues Yacine Challal and Imed Romdhani from Compiegne University for their help during our stay at Heudiasyc. Many thanks are due to our colleagues at CERIST, Souad Benmeziane and Derhab Abdelouahid, for their reviews. Special thanks to the anonymous referees for their comments and suggestions, which considerably helped us to improve this article.

References

[1] W. Stallings, *Cryptography and Network Security Principles and Practices*, 3rd ed., Pearson Education Inc., 2003.
[2] F. Stajano and R. Anderson, "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks," *7th Int'l. Security Protocols Wksp.*, Cambridge, UK, April 1999.
[3] The IETF Web site, http://www.ietf.org
[4] B. David and A. David, "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing*, Chapter 5, pp. 153–81, 1996.
[5] Y.-B. Ko and N. Vaidya, "Location-aided routing, LAR, in mobile ad hoc networks," *ACM/IEEE MOBICOM'98*, Dallas, Texas, October 1998, pp. 66-75.

[6] V. D. Park and M. S. Corson, "A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks," *INFOCOM '97*, Apr. 1997, pp. 1405–13.
[7] C. Perkins and E. Royer, "Ad Hoc on Demand distance Vector (AODV) Algorithm," *2nd IEEE Wksp. Mobile Comp. Systems and Applications (WML'SA'99)*, 1999, pp. 90–100.
[8] C.-K. Toh, "A novel Distributed Routing Protocol to Support Ad Hoc Mobile Computing," *IEEE 15th Annual Int'l. Phoenix Conf. Comp. and Commun.*, Mar. 1996, pp. 480–86.
[9] C. Perkins and R. Bhagwat, "Highly Dynamic Destination-sequenced Distance-vector Routing (DSDV) for Mobile Computer," *ACM SIGCOMM'94 Conf. Commun. Architectures, Protocols and Applications*, Mar. 1994, pp. 234–44.
[10] S. Murthy and J. J. Garcia-Luna-Aceves, "An Efficient Routing Protocol for Wireless Networks," *ACM Mobile Networks and Application J.*, special issue on *Routing in Mobile Commun. Networks*, Oct. 1996, pp. 183–97.
[11] E. Royer and C. Toh, "A Review of Current Routing Protocols for Ad-hoc Mobile Wireless Networks," *IEEE Pers. Commun. Mag.*, vol. 6, Apr. 1999, pp. 46–55.
[12] N. Badache, D. Djenouri, and A. Derhab, "Mobility Impact on Mobile Ad Hoc Networks," *ACS/IEEE Conf. Proc.*, Tunis, Tunisia, July 2003.
[13] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Ariadne: A Secure On-demand Routing Protocol for Ad Hoc Networks," *8th Annual Int'l. Conf. Mobile Comp. and Net. MobiCom 2002*, Sept. 2002, pp. 12–23.
[14] Y.-C. Hu, D. B. Johnson, and A. Perrig, "Secure Efficient Distance Vector Routing in Mobile Wireless Ad Hoc Networks," *4th IEEE Wksp. Mobile Computing Systems and Applications WMCSA '02*, June 2002.
[15] Y.-C. Hu, D. Johnson, and A. Perrig, "SEAD: Secure Efficient Distance Vector Routing for Mobile Wireless Ad Hoc Networks," *Ad Hoc Networks* (Elsvier), vol. 1, no. 1, 2003, pp. 175–92.
[16] C. Castelluccia and G. Montenegro, "Protecting AODV Against Impersonation Attacks," *ACM SIGMOBILE Mobile Comp. and Commun. Rev. Archive*, vol. 6, no. 3, July 2002, pp. 108–09.
[17] P. Papadimitratos and Z. J. Haas, "Secure Routing For Mobile Ad Hoc Networks," *SCS Commun. Net. and Distributed Systems Modeling and Simulation Conf. (CNDS 2002)*, Jan. 2002.
[18] A. Perrig *et al.*, "Spins: Security Protocols for Sensor Networks," *7th Annual ACM Int'l. Conf. Mobile Computing and Networks (MobiCom 2001)*, Rome, Italy, July 2001.
[19] K. Sanzgiri *et al.*, "A Secure Routing Protocol for Ad Hoc Networks," *10th IEEE Int'l. Conf. Network Protocols (ICNP '02)*, Nov. 2002.
[20] M. G. Zapata and N. Asokan, "Securing Ad Hoc Routing Protocols," *ACM Wksp. Wireless Security (WiSe 2002)*, Sept. 2002.
[21] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Rushing Attacks and Defense in Wireless Ad Hoc Network Routing Protocols," *ACM Wksp. Wireless Security WiSe 2003*, San Diego, CA, USA, Sept. 2003.
[22] S. Yi, R Naldurg, and R. Kravets, "Security-aware Ad-hoc Routing for Wireless Networks," *ACM Wksp. Mobile Ad Hoc Networks*, Mobihoc, 2001.
[23] P. Papadimitratos and Z. Haas, "Secure Routing for Mobile Ad Hoc Networks," *SCS Commun. Net. and Distributed Systems Modeling and Simulation Conf. CNDS*, San Antonio, Texas, Jan. 2002.
[24] Y.-C. Hu and A. Perrig, "A Survey of Secure Wireless Ad Hoc Routing," *IEEE Security and Privacy*, vol. 2, no. 3, 2004, pp. 28–39.
[25] F. Wang, B. Vetter, and S. Wu, "Secure Routing Protocols: Theory and Practice," North Carolina State University, Tech. Rep., May 1997.
[26] Y.-C. Hu, A. Perrig, and D. B. Johnson, "Packet Leashes: A Defense Against Wormhole Attacks in Wireless Ad Hoc Networks," *22nd Annual Joint Conf. IEEE Comp. and Commun. Societies (INFOCOM 2003)*, Apr. 2003.
[27] B. Awerbuch *et al.*, "An On Demand Secure Routing Protocol Resilient to Byzantine Failures," *ACM Wksp. Wireless Security (WiSe)*, Atlanta, Georgia, Sept. 2002.
[28] D. Djenouri and N. Badache, "New Power-aware Routing for Mobile Ad Hoc Networks," accepted in the *Int'l. J. Ad Hoc and Ubiquitous Computing (Inderscience)*, vol 1, no. 3, 2005.
[29] P. Papadimitratos and Z. J. Haas, "Secure Data Transmission in Mobile Ad Hoc Networks," *2003 ACM Wksp. Wireless Security*, San Diego, CA, USA, 2003, pp. 41–50.
[30] S. Marti *et al.*, "Mitigating Routing Misbehavior in Mobile Ad Hoc Networks," *ACM Mobile Comp. and Net., MOBICOM 2000*, pp. 255–65.
[31] S. Doshi and T. Brown, "Minimum Energy Routing Schemes for a Wireless Ad Hoc Network," *IEEE INFOCOM'02*, New York City, USA, 23–27 June 2002.
[32] D. Djenouri and N. Badache, "Simulation Performance Evaluation of an Energy Efficient Routing Protocol for Mobile Ad Hoc Networks," *IEEE/ACS Int'l. Conf. Pervasive Services (ICPS'04)*, American University of Beirut (AUB), Lebanon, July 2004.
[33] F. Kargl *et al.*, "Advanced Detection of Selfish or Malicious Nodes in Ad Hoc Networks," *1st European Wksp. Security in Ad-Hoc and Sensor Networks*, ESAS 2004, Aug. 5–6, 2004.
[34] H. Yang, X. Meng, and S. Lu, "Self-organized Network Layer Security in Mobile Ad Hoc Networks," *ACM MOBICOM Wireless Security Wksp (WiSe'02)*, Sept. 2002.
[35] A. Shamir, "How to Share A Secret," *Commun. ACM*, vol. 22, no. 11, Nov. 1979, pp. 612–13.
[36] P. Z. J. Kong, H. Luo, S. Lu, and L. Zhang, "Providing Robust and Ubiquitous Security Support for Manet," *9th IEEE Int'l. Conf. Network Protocols ICNP 2001*, Nov. 2001, pp. 251–60.
[37] P. Michiardi and R. Molva, "Core: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks," *Commun. and Multimedia Security 2002 Conf.*, Portoroz, Slovenia, Sept. 26–27 2002.
[38] S. Buchegger and J.-Y. L. Boudec, "A Robust Reputation System for Mobile Ad-hoc Networks," EPFL IC, Tech. Rep. IC/2003/50, July 2003.
[39] S. Buchegger and J. LeBoudec, "Performance Analysis of the Confidant, Protocol Cooperation of Nodes Fairness in Dynamic Ad Hoc Networks," *3rd ACM Int'l. Symp. Mobile Ad Hoc Networking and Computing (MobiHoc '02)*, Lausanne, Switzerland, June 2002, pp. 80–91.
[40] S. Buchegger and J.-Y. Le-Boudec, "A Robust Reputation System for p2p and Mobile Ad-hoc Networks," *2nd Wksp. Economics of Peer-to-Peer Systems*, June 2004.
[41] M. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance," *J. ACM*, vol. 36, no. 2, Apr. 1989, pp. 335–48.
[42] J.-P. Hubaux *et al.*, "Towards Self-organized Mobile Ad Hoc Networks: the Terminodes Project," *IEEE Commun. Mag.*, Jan. 2001.
[43] L. Buttyan and J.-P. Hubaux, "Nuglets: A Virtual Currency to Stimulate Cooperation in Self-organized Mobile Ad Hoc Networks," Swiss Federal Institution of Technology, Lausanne, Switzerland, Tech. Rep. DSC/2001/001, Jan. 2001.
[44] L. Buttyn and J.-P. Hubaux, "Stimulating Cooperation in Self-organizing Mobile Ad Hoc Networks," *ACM/Kluwer Mobile Net. and Applications (MONET)*, vol. 8, no. 5, Oct. 2003.
[45] L. Buttyan and J. Hubaux, "Stimulating Cooperation in Self-organizing Mobile Ad Hoc Networks," *ACM/Kluwer Mobile Networks and Applications*, vol. 8, no. 5, Oct. 2003.
[46] R. Jurdak, C. V. Lopes, and P. Baldi, "A Survey, Classification and Comparative Analysis of Medium Access Control Protocols for Ad Hoc Networks," *IEEE Commun. Surveys and Tutorials*, vol. 6, no. 1, 2004.
[47] M. S. Gast, *802.11 Wireless Networks*, 1st ed. O'Reilly and Association, mc, 2002.
[48] P. Kyasanur and N. H. Vaidya, "Detection and Handling of MAC Layer Misbehavior in Wireless Networks," *IEEE Int'l. Conf. Dependable Systems and Networks (DSN'03)*, San Francisco, California, Jun 2003.
[49] T.-C. Chiang and Y-M. Huang, "Group Keys and the Multicast Security in Ad Hoc Networks," *1st Int'l. Wksp. Wireless Security and Privacy (WiSPr'03)*, 2003.
[50] R. di Pietro *et al.*, "A Directed Diffusion-based Secure Multicast Scheme for Wireless Sensor Networks," *1st Int'l. Wksp. Wireless Security and Privacy (WiSPr'03)*, 2003.

[51] L. Lazos, R. Poovendran, and G. H. Cirincione, "Location-aware Secure Wireless Multicast in Ad-hoc Networks Under Heterogeneous Path-loss," University of Washington, Electrical Engineering Department, Tech. Rep. UWEETR-2003-0012, 2003.
[52] A. Yasinsac and J. Davis, "Modeling Protocols for Secure Group Communication in Ad Hoc Networks," *10th Int'l. Wksp. Security Protocols*, Cambridge, UK, Apr. 2002.
[53] S. Maki, T. Aura, and M. Hietalahti, "Robust Membership Management for Ad-hoc Groups," *5th Nordic Wksp. Secure IT Systems (NORDSEC'2000)*, 2000.
[54] M. Steiner, G. Tsudik, and M. Waidner, "A New Approach to Group Key Agreement," *Int'l. Conf. Distributed Computing Systems*, 1998, pp. 380–87.
[55] W. Diffie and M. E. Hellman, "New Directions in Cryptography," *IEEE Trans. Info Theory*, vol. IT-22, no. 6, 1976, pp. 644–54; available at: citeseer.ist.psu.edu/diffie76new.html
[56] M. Steiner, G. Tsudik, and M. Waidner, "Diffie Hellman Key Distribution Extended to Group Communication," *ACM Conf. Comp. and Commun. Security*, 1996, pp. 31–37.
[57] S. M. Bellovin and M. Merritt, "Encrypted Key Exchange: Password-based Protocols Secure Against Dictionary Attacks," *IEEE Symp. Security and Privacy*, May 1992, pp. 72–84.
[58] N. Asokan and Rinzboorg, "Key Agreement in Ad Hoc Networks," *Comp. Commun.*, vol. 23, no. 17, 2000, pp. 1627–37.
[59] K. Becker and U. Wille, "Communication Complexity of Group Key Distribution," *5th ACM Conf. Comp. and Commun. Security*, 1998, pp. 1–6.
[60] X.-Y. Li, Y. Wang, and O. Frieder, "Efficient Hybrid Key Agreement Protocol for Wireless Ad Hoc Networks," *IEEE Int'l. Conf. Comp. Commun. and Networks ICCCN'02*, Miami, FL, USA, 2002.
[61] S. Zhu, S. S. S. Xu, and S. Jajodia, "Gkmpan: An Efficient Group Rekeying Scheme for Secure Multicast in Ad-Hoc Networks," *1st Annual Int'l. Conf. Mobile and Ubiquitous Systems: Net. and Services (MobiQuitous'04)*, Aug. 22–26 2004, pp. 42–51.
[62] C. K. Wong, M. G. Gouda, and S. S. Lam, "Secure Group Communications Using Key Graphs," *ACM SIGCOMM '98 Conf. Applications, Technologies, Architectures, and Protocols for Comp. Commun.*, 1998, pp. 68–79.
[63] A. Perrig, D. X. Song, and J. D. Tygar, "ELK, A New Protocol for Efficient Large-group Key Distribution?" IEEE Symp. Security and Privacy, 2001, pp. 247–62.
[64] D. Naor, M. Naor, and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 9, no. 43, 2002.
[65] D. Liu, P. Ning, and K. Sun, "Efficient Self-healing Group Key Distribution with Revocation Capability," *10th ACM Conf. Comp. Commun. Security CCS*, Washington DC, USA, Oct. 27–30 2003, pp. 231–40.
[66] J. Staddon *et al.*, "Self-healing key Distribution with Revocation," *IEEE Symp. Security and Privacy*, May 2002.
[67] L. Lazos and R. Poovendran, "Energy-aware Secure Multicast Communication in Ad-hoc Networks Using Geographic Location Information," *IEEE Int'l. Conf. Acoustics Speech and Sig. Proc.*, Hong Kong, China, Apr. 6–10 2003.
[68] T. Kaya *et al.*, "Secure Multicast Groups on Ad Hoc Networks," *1st ACM Wksp. Security of Ad hoc and Sensor Networks SASN'03*, Fairfax, VA, USA, Oct. 2003.
[69] S. Das, B. Manoj, and C. Murthy, "A Dynamic Core Based Multicast Routing Protocol for Ad Hoc Wireless Networks," *Mobile Ad Hoc Networks Wksp. (MADNET)*, Sophia-Antipolis, France, 2003.
[70] T. Ozaki, J. Kim, and T. Suda, "Bandwidth Efficient Multicast Routing Protocol for Ad Hoc Networks," *IEEE ICCCN*, Oct. 1999, p. 10 17.
[71] E. Royer and C. Perkins, "Multicast Using Ad Hoc on Demand Distance Vector Routing," *ACM MOBICOM*, 1999, pp. 207–18.
[72] P. Sinha, S. Sivakumar, and V. Bharghavan, "MCEDAR: Multicast Core Extraction Distributed Ad Hoc Routing," *IEEE WCNC*, Aug. 1999, p. 1313–17.
[73] J. Jetcheva and D. Johnson, "Adaptive Demand-driven Multicast Routing in Multi-hop Wireless Ad Hoc Networks," *MOBIHOC*, 2001, p. 33–44.
[74] S. Lee, M. Gerla, and C. Chiang, "On Demand Multicast Routing Protocol," *IEEE WCNC*, Aug. 1999, pp. 1298–302.
[75] J. Garcia-Luna-Aceves and E. Madruga, "The Core-assisted Mesh Protocol," *IEEE JSAC*, vol. 17, no. 8, 1999, pp. 1380-1994.
[76] S. Lee and C. Kim, "Neighbor Supporting Ad Hoc Multicast Routing Protocol," *ACM MOBIHOC*, Aug. 2000, pp. 37–50.
[77] G. Lin and G. Noubir, "Secure Multicast over Multihop Wireless Ad Hoc Networks," *ACM MOBIHOC*, June 2002.
[78] L. Zhou and Z. J. Haas, "Securing Ad Hoc Networks," *IEEE Network*, vol. 13, no. 6, 1999, pp. 24–30.
[79] R. Ostrovsky and M. Yung, "How to Withstand Mobile Virus Attacks," *10th ACM Annual Symp. Principles of Distributed Computing (PODC9I)*, Montreal, Quebec, Canada, Augest 1991, pp. 51–59.
[80] S. Yi and R. Kravets, "Moca : Mobile Certificate Authority for Wireless Ad Hoc Networks," *2nd Annual PKI Research Wksp. (PKI '03)*, Gaithersburg, MD, 2003.
[81] S. Capkun, L. Buttyan, and J.-P. Hubaux, "Self-organized Public Key Management for Mobile Ad Hoc Networks," *IEEE Trans. Mobile Comp.*, vol. 2, no. 1, Jan. 2003, pp. 52–64.
[82] J.-P. Hubaux, L. Bunyan, and S. Capkun, "The Quest for Security in Mobile Ad Hoc Networks," *MobiHoc '01: Proc. 2nd ACM Int'l. Symp. Mobile Ad Hoc Net. and Comp.*, ACM Press, 2001, pp. 146–55.
[83] P. Zimmermann, *The Official PGP Users Guide*, MIT Press, 1995.
[84] R. Heady *et al.*, "The Architecture of a Network Level Intrusion Detection System," Computer Science Department, University of New Mexico, Tech. Rep., Aug. 1990.
[85] Y. H. W. Lee, "A Cooperative Intrusion Detection System for Ad Hoc Networks," *1st ACM Wksp. Security of Ad Hoc and Sensor Networks*, Fairfax, Virginia, USA, 2003, pp. 135–47.
[86] Y. Zhang, W. Lee, and Y. Huang, "Intrusion Detection Techniques for Mobile Wireless Networks," *ACM Wireless Networks*, vol. 9, no. 5, Sept. 2003, pp. 545–56.
[87] T. Lunt *et al.*, "A Real-time Intrusion Detection Expert System (ides)," Computer Science Laboratory, SRI International, Menlo Park, California, Tech. Rep., Feb. 1992.
[88] K. Ilgun, R. A. Kemmerer, and P. A. Porras, "State Transition Analysis: A Rule-based Intrusion Detection Approach," *IEEE Trans. Software Eng.*, vol. 21, no. 3, Mar. 1995, pp. 181–99.
[89] S. Kumar and E. H. Spafford, "A Software Architecture to Support Misuse Intrusion Detection," *18th National Info. Security Conf.*, 1995, pp. 194–204.
[90] F. Anjum, D. Subhadrabandhu, and S. Sarkar, "Signature based Intrusion Detection for Wireless Ad-hoc Networks: A Comparative Study of Various Routing Protocols," *Vehic. Tech. Conf., Wireless Security Symp.*, Orlando, Florida, USA, Oct. 2003.
[91] Y. Huang *et al.*, "Cross-feature Analysis for Detecting Ad-hoc Routing Anomalies," *23rd Int'l. Conf. Distributed Comp. Sys.*, Providence, RI, May 2003.
[92] O. Kachirski and R. Guha, "Effective Intrusion Detection Using Multiple Sensors in Wireless Ad Hoc Networks," *36th Annual Hawaii Int'l. Conf. System Sciences (HICSS'03)*, Big Island, Hawaii, Jan. 2003.
[93] I. F. Akyildiz *et al.*, "Wireless Sensor Networks: A Survey," *Computer Networks: The Int'l. J. Comp. and Telecommun. Net.*, vol. 38, no. 4, 2002, citeseer.ist.pso.edo/diffie76new.html, pp. 393–422.
[94] N. Xu, "A Survey of Sensor Network Applications," University of Southern California, Tech. Rep., 2002.
[95] A. D. Wood and J. A. Stankovic, "Denial of Service in Sensor Networks," *Computer*, vol. 35, no. 10, 2002, pp. 54–62.
[96] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasure," *1st IEEE International Workshop on Sensor Network Protocols and Applications*, USA, 2003.
[97] C. Intanagonwiwat, R. Govindan, and D. Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks," *ACM MobiCom'00*, 2001, pp. 2009–15.
[98] R. C. Merkle, "Protocols for Public Key Cryptosystems," *IEEE Symp. Research in Security and Privacy*, Apr. 1980.

[99] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy Efficient Communication Protocol for Wireless Micro Sensor Networks," *Int'l. Conf. Parallel Proc.*, 2001, pp. 156–63.
[100] L. Eschenauer and V. D. Gligor, "A Key Management Scheme for Distributed Sensor Networks," *CC502*, Washington DC, USA, 2002.
[101] P. Ganesan *et al.*, "Analyzing and Modeling Encryption Overhead for Sensor Network Nodes," *WSNA '03: the 2nd ACM Int'l. Conf. Wireless Sensor Networks and Applications,* New York, N1 USA: ACM Press, 2003, pp. 151–59.
[102] R. Venugopalan *et al.*, "Encryption Overhead in Embedded Systems and Sensor Network Nodes: Modeling and Analysis," *CASES '03: the 2003 Int'l. Conf. Compilers, Architecture and Synthesis for Embedded Systems*, San Jose, California, USA, 2003, pp. 188–97.
[103] G. G. Xie, C. E. Irvine, and T. E. Levin, "Quantifying Effect of Network Latency and Clock Drift on Time-driven Key Sequencing," *ICDCSW '02: Proc. 22nd Int'l. Conf. Distributed Computing Systems*, Washington, DC, USA: IEEE Comp. Society, 2002, pp. 35–42.
[104] A. Manjeshwar and D. R. Agrawal, "Teen: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks," *15th Parallel and Distributed Proc. Symp.*, Boston, MA, 2000, p. 56–67.
[105] J. M. Kahn, R. H. Katz, and K. S. J. Pister, "Next Century Challenges: Mobile Networking for "Smart Dust,"" *Int'l. Conf. Mobile Computing and Net. (MOBICOM)*, 1999, citeseer.ist.psu.edu/kahn99next.html, pp. 271–78.
[106] D. W. Carman, R S. Kruus, and B. J. Mall, "Constraints and Approaches for Distributed Sensor Network Security," NAI Labs, Tech. Rep. 00-010, 2000.
[107] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks," *IEEE Symp. Research in Security and Privacy*, 2003.
[108] S. Zhu *et al.*, "Establishing Pair-wise Keys for Secure Communication in Ad Hoc Networks: A Probabilistic Approach," *11th IEEE Int'l. Conf. Network Protocols*, 2003.
[109] R. D. Pietro, L. Mancini, and A. Mci, "Efficient and Resilient Key Discovery based on Pseudo-random Key Pre-deployment," *18th Int'l. Parallel and Distributed Processing Symp.*, Apr. 2004.
[110] G. Gaubatz, J.-P. Kaps, and B. Sunar, "Public Key Cryptography in Sensor Networks Revisited," *1st European Wksp. Security in Ad-Hoc and Sensor Networks (ESAS 2004)*, 2004.
[111] J. Hofflein, J. Pipher, and J. Silverman, "Ntru: A Ring-based Public Key Cryptosystem," *Algorithmic Number Theory (ANTS III)*, vol. 1423 of LNCS, 1998, pp. 267–88.
[112] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," *Elsevier's Ad Hoc Networks Journal*, special issue on sensor network applications and protocols, 2002.
[113] Deng, R. Han, and S. Mishra, "Insens: Intrusion-tolerant Routing in Wireless Sensor Networks," *23rd IEEE Int'l. Conf. Distributed Comp. Sys. (ICDCS 2003)*, May 2003.
[114] J. Deng, R. Han, and S. Mishra, "A Performance Evaluation of Intrusion-tolerant Routing in Wireless Sensor Networks," IPSN, 2003, pp. 349–64.
[115] W. Du *et al.*, "A Witness-based Approach for Data Fusion Assurance in Wireless Sensor Networks," *GLOBECOM '03*, 2003.
[116] B. Przydatek, D. Song, and A. Perrig, "SIA: Secure Information Aggregation in Sensor Networks," *SenSys03*, 2003.
[117] S. R. Madden *et al.*, "Tag: A Tiny Aggregation Service for Ad-hoc Sensor Networks," *OSDI*, Dec. 2002.
[118] J. Deng, R. Han, and S. Mishra, "Security Support for In-network Processing in Wireless Sensor Networks," *2003 ACM Wksp. Security of Ad Hoc and Sensor Networks (SASN '03)*, Oct. 2003.

BIOGRAPHIES

DJAMEL DJENOURI (ddjenouri@mail.cerist.dz) obtained the engineer degree in computer science and the master degree in computer science from the University of Science and Technology USTHB (Algiers) in 2001 and 2003, respectively. He is currently a Ph.D. student at USTHB and a research assistant at the CERIST Center of Research in Algiers. He works mainly on ad hoc networking, especially in the areas of security, power management, routing protocols, and MAC protocols.

LYES KHELLADI (lkhelladi@mail.cerist.dz) received the engineer degree in computer science from the University of Science and Technology USTHB (Algiers) 2001. He is currently working toward his PhD at USTHB, and he is a research assistant at the CERIST Center of Research in Algiers. His main topics of interest include sensor networks, computer security, energy efficient communication schemes, routing protocols, and MAC protocols.

NADJIB BADACHE (badache@wissal.dz) is a professor in the computer science department of USTHB University, where he is also the head of the LSI laboratory. He is an author of many papers, and he has supervised many thesis and research projects. His areas of interest are distributed mobile systems, mobile ad hoc networks, and security.