LOCAL MANAGEMENT OF CREDITS AND DEBITS IN MOBILE AD HOC NETWORKS*

Fabio Martinelli, Marinella Petrocchi, and Anna Vaccarelli
Istituto di Informatica e Telematica - CNR - Pisa, Italy
{ fabio.martinelli,marinella.petrocchi,anna.vaccarelli } @iit.cnr.it

Abstract Nodes in mobile ad hoc networks often need the help of others in order to have their packets delivered to their destination. However, nodes may not be governed by a single authority and need not share a common goal. Thus, a *selfish* node may prefer to save resources for its own communication, rather than to forward packets for other nodes. We suggest to collect information about the forwarding of packets in the network in a decentralized manner. Through reception of acknowledgments, a node can update a local repository, on which the node can rely to judge the behavior of the other nodes. We define a secure structure for the acknowledgments and the rules for updating the local repository. Also, we discuss a solution to achieve a univocal identification of a node in MANET environments.

Keywords: Mobile Ad Hoc Networks (MANETs), Selfishness

1. Introduction

Unlike traditional mobile networks, ad hoc networks do not rely on any wired infrastructure. Instead, the network is kept connected by the mobile hosts. In order to make a mobile network functional, the nodes need to be self-organized, in such a way that a message is delivered from a source to a destination through a set of intermediate nodes. The deployment of ad hoc networks for civilian applications is taking a footing. In such applications, the nodes are not governed by a single authority and need not share a common goal (the contrary could be the case in emergency and military applications). Thus, cooperative behaviors, such as forwarding each other's packets, cannot be easily assumed. The single nodes could prefer to save battery life for their own communication, rather than to forward packets for other nodes. Such an attitude is denoted in the recent literature as *selfishness* of the node. Simulation results (see, *e.g.,* (Michiardi and Molva, 2002b)) have recently pointed out that

*Work partially supported by the MIUR project: "Strumenti, Ambienti e Applicazioni Innovative per la Società dell'Informazione", sottoprogetto SP1: Reti INTERNET: "efficienza, integrazione e sicurezza"; by the CSP project: SeTAPS II; by the Quality of Protection (QoP) project: CREATE-NET.

a selfish behavior can be as harmful, in terms of the network throughput, as a malicious one.

There is a growing interest in the research community for detecting and preventing selfish behavior, and promoting cooperation between nodes, (see, *e.g.*, (Buchegger and Boudec, 2002; Buttyan and Hubaux, 2002; Marti et al., 2000; Michiardi and Molva, 2002a; Salem et al., 2003; Zhong et al., 2003)). Here, we propose an infrastructure for a local management of credits (*i.e.*, a measure of how many packets node A has forwarded for node B) and debits (*i.e.*, a measure of how many packets node B has forwarded for node A). Each node maintains this information in a local repository that we call *credit table*, on which the node can rely to judge the past behavior of the other nodes in the network. More specifically, we define rules for the table initialization, its maintenance, and secure acknowledgments testifying the actual forwarding of packets in the network.

MANETs are prone to the following security threat: a node could be tempted to discard its initial identity and re-enter the network in disguise in environments where i) users are punished for their selfish behavior, or ii) new users are *a priori* granted to have an initial amount of packets forwarded. We investigate solutions to achieve a univocal relation between a physical device and the identity it claims at its first steps in the network.

We propose to use network-layer acknowledgments (additional data in routing protocols specifications like (Johnson et al., 2001)) to provide to the packet source an authenticated proof that the packet has been delivered to its destination. We specify the structure for the acknowledgment request and the corresponding acknowledgment. Then, we introduce a mechanism that amortizes the signaling of "occurred delivery" over blocks of *n* data packets, thus reducing the communication overhead on the way back from destination to source. Further, we deal with some kind of attacks to which our scenario is prone.

Finally, in the model we have developed, if node A behaves well in forwarding packets for node B, then it can exploit this correct behavior only with B (meaning that A may rely on routes including B for sending its packets). Intuitively, systems based on such a rule can get stuck. Then we introduce the notion of credits *transferring*, according to which A may ask B to transfer, in a secure way, its credits to some other nodes.

The remainder of the paper is organized as follows: the next section illustrates related work in the area. In Section 3, we summarize the basic operation of the Dynamic Source Routing protocol (DSR), on which we rely at routing level. In Section 4 we define the trust setup of the network and the adversary model. Section 5 is dedicated to design the structure of the *credit table*. Then, we introduce the notion of credit *transferring* and we conclude the paper.

2. Related Work

We discuss here some work related to secure on demand routing protocols and cooperation enforcement in mobile ad hoc networks.

So called –on demand routing protocols– are those routing protocols in which a node tries to discover a route only when it has a packet to send.

Among on demand routing protocols, the Dynamic Source Routing protocol (DSR (Johnson et al., 2001)) is a protocol providing self-organization in configuring routing topologies for mobile wireless networks. It consists of two main phases, Route Discovery, the mechanism by which a source node, that does not have in its *Route Cache* the route to some destination yet, initiates to find a route, and Route Maintenance, the mechanism by which the source node detects, while sending a packet to some destination, if the route has been broken.

Hu *et al.* propose Ariadne, (Hu et al., 2002), securing a basic version of DSR. Ariadne provides: i) source authentication at the target's side; ii) authentication of each entry of the discovered path at the source's side; iii) integrity of the discovered path.

In (Zhou and Haas, 1999), the authors highlight peculiarities of ad hoc networks to fight against possible misbehaviors. Since routing protocols like DSR can return multiple routes, a node could exploit this *redundancy* to switch to an alternative route when the primary one has been broken because of a misbehavior.

A reputation system may be used in ad hoc networks to provide incentives in order to forward messages, *e.g.,* see (Buchegger and Boudec, 2002; Marti et al., 2000; Michiardi and Molva, 2002a). Both (Buchegger and Boudec, 2002) and (Marti et al., 2000) assume a network layer based on DSR. In (Marti et al., 2000), the authors consider complementing DSR with a watchdog mechanism to identify the misbehaving nodes, plus a path-rater mechanism to build new routes avoiding those nodes. Even if they show it is possible to keep the throughput of the network over a certain threshold even in presence of misbehaving nodes, the last are still allowed to send and receive packets. In (Buchegger and Boudec, 2002), the authors choose to act in a similar manner. They propose the CONFIDANT protocol, in which DSR is fortified by a neighborhood monitoring[1] and a trust manager which sends and receives alarm messages to and from other trust managers. A reputation system maintains a table listing ratings for all nodes and a path manager changes the route when the ratings for some nodes fall under a certain threshold. Hence, misbehaving nodes are totally isolated from the rest of the network.

We base our work on a (secure) DSR, like (Buchegger and Boudec, 2002; Marti et al., 2000) for MANETs. Similar to (Marti et al., 2000), our nodes do not exchange information with each other and they locally maintain history about their past behavior. Contrary to (Marti et al., 2000), we achieve information through cryptographic and acknowledgment mechanisms, whereas (Marti et al., 2000) assumes wireless interfaces that support promiscuous mode oper-

[1] In broadcast mediums, hosts are able to listen to messages that are not addressed to them. In particular, neighboring nodes are able to listen to their next-hop node transmissions.

ation. When this mode is enabled, a node can listen in on a neighbor's traffic. Thus, when A forwards a packet to B, A can overhear if B, in its turn, forwards the packet. Hence, (Marti et al., 2000) relies on first-hand information (*e.g.*, experienced and observed forwarding behavior of neighbors). Instead, we rely on trusted second-hand information, close to the approach of (Buchegger and Boudec, 2002), but we do not directly punish misbehaved nodes, rather we distinguish the well-behaved ones. Further, we focus on cryptographic solutions to handle the security of the information about the attitude of the nodes w.r.t. forwarding or dropping packets.

Michiardi and Molva, (Michiardi and Molva, 2002a), analyze enforcement of cooperation in game theoretical terms. The authors introduce the concept of redemption of nodes, *i.e.*, a misbehaving node starts well-behaving can be re-integrated in the network. The work in (Urpi et al., 2003) develops a formal model, based on game theory too, that captures features of MANETs like node mobility and selfishness. The paper provides a general model to describe cooperation enforcement policies.

Another possibility to provide incentives is to award well-behaving nodes with credits. (Buttyan and Hubaux, 2002) introduces a virtual currency called *nuglets*, by which a node is being paid when it forwards packets. Also, the node is forced to pay nuglets to send its own packets. With a pure selfish behavior, the node will soon finish its money and will not be able to send packets. In order to avoid the possibility that a node arbitrarily increases its own nuglets, a tamper-proof security module is required at each node.

The approach in (Buttyan and Hubaux, 2002) may appear close to our approach. As shown in Section 5, our packet source increases a debit counter for B upon receiving a proof that B has actually forwarded the source's packets. On the other hand, (Buttyan and Hubaux, 2002)'s philosophy is different from ours since our money is not physically gained by B, rather we rely on the fact that the source reasonably returns the favor to B for subsequent communication. Further, we do not put constraints to the node's capability to send packets. For this reason, and for the fact that each node will base its behavior on the data locally maintained, we do not need a tamper-proof module at each node. Within our framework, the credits that we gain cannot be spent with all nodes in the network, but only with those nodes for which we have forwarded something. On the contrary, *nuglets* can be spent for sending packets over all the available routes. We try to fill this gap by introducing the notion of credit *transferring*.

An award-based technique has been recently proposed also in (Zhong et al., 2003), where the authors rely on a central authority. Basically, when a node receives a message, it keeps a receipt for that message. Then, the node reports to the authority all the collected receipts. The authority evaluates the receipts and, consequently, it assigns charges and credits. The system does not need tamper-proof modules. (Zhong et al., 2003) presents similarities with our work because it considers secure receipts to testify the correct packet delivery. However, we rely on a central infrastructure only when a node enters the network, in order to bootstrap trust. Indeed, our solution exactly tries to avoid the necessity of such a central authority during the whole lifecycle of the community. We propose a self-organized credit management.

We ought to cite relevant work related to enforce cooperation between nodes belonging to other scenarios. Indeed, besides pure ad hoc networks, so called multi-hop cellular networks are getting a footing too. They combine features of both cellular and mobile ad hoc networks. Basically, they are cellular networks where there is the possibility of peer to peer or relayed multi-hop connections. Mobile hosts communicate with a wired infrastructure by means of wireless technology. A peculiarity is that communication between a base station and a mobile station may be relayed by other mobile stations. As novel work on cooperation in multi-hop cellular networks, we cite the approach of (Salem et al., 2003). Here, all communication between mobile hosts are required to pass through a base station, that actually acts as an authority for the distribution of symmetric primitives for securing data. Further, the base station is responsible for charging the initiator of a communication and for awarding the forwarding nodes.

Note that a multi-hop cellular network scenario allows (Salem et al., 2003) to exploit a base station either for the distribution of secret keys, thus exploiting symmetric cryptography between the base station and the nodes, and for charging and awarding nodes. Thus, (Salem et al., 2003) nicely addresses a scenario where a central authority is given for free.

(Lamparter et al., 2003) proposes another award-based mechanism for motivating cooperation in what the authors call *stub* ad hoc networks, *i.e.*, mobile networks with access to the Internet. Again, an external third party authenticates the nodes involved in a communication and it assigns charges and credits.

3. DSR

DSR (Dynamic Source Routing, (Johnson et al., 2001)) is an on-demand routing protocol designed to be used in mobile ad hoc networks. It consists of two main phases, Route Discovery, *i.e.*, the mechanism by which a source node initiates to find a route, and Route Maintenance, the mechanism by which the source node detects, while sending a packet to some destination, if the route has been broken.

The initiator of Route Discovery sends a Route Request message as a local broadcast specifying the Discovery's target. Each node receiving the request appends to the request its own IP address, unless it has recently seen that request, then it re-broadcasts the request. When the target receives the request, it creates a Route Reply message containing the list of addresses and sends it back to the initiator.

Route Maintenance monitors the reliability of a route. Detection of link breaks is often provided at no cost, when the routing protocol in use relies on a Medium Access Control protocol such as 802.11 (of the IEEE Computer Society, 1999), that provides link-layer acknowledgments. In this case, to test the reachability of the next-hop node, the previous-hop node waits for the reception of a link-layer acknowledgment (ACK). A limited number of retransmissions of the same packet is due, then, if the node does not receive link-layer ACKs from its next-hop neighbor, it sends a Route Error message back to the source, notifying it of a link break.

Instead of using link-layer acknowledgments, a node can explicitly require a network-layer acknowledgment to the next-hop neighbor (Johnson et al., 2001). The acknowledgment request is added as an optional part in the DSR header. In Section 5, we will propose to exploit network-layer ACKs to convey information about the actual forwarding of packets in the route. Though these ACKs were born with the intent of detecting link failures, we will exploit them to update information that each node locally maintains. We will suitably modify the acknowledgment mechanism, such that the nodes will be able to prove to have forwarded packets along a certain route.

Hereafter, it is understood that we rely on link layer ACKs at Medium Access Control level to detect link failures, while network-layer ACKs are used to convey information about the forwarding of packets. Further, we assume bidirectional communication on every link, *i.e.*, if node A is able to transmit to node B, then B is able to transmit to A.

4. Trust Setup

Before deployment, each node in the network generates a pair of public/private keys. A correct use of asymmetric cryptography requires to authenticate in a secure manner the association between the public keys and the identities with which they are associated. Public key certificates are a very well-known solutions to manage the matter. In frameworks where certificates validate the nodes at stake, we get into the issue of defining to which identifier (*e.g.*, node's identifier, IP address, MAC address, *etc.*) a public key must be associated. Indeed, an user could be tempted to discard its initial identity (hence requiring a new certificate, tied to a new identity) when its rating falls below a certain threshold. In reputation systems where misbehaving nodes are punished according to their reputation ratings, *e.g.*, by being isolated from the rest of the network, a way for the node to re-enter the network is to start from the beginning in disguise. Thus, the key point for MANETs is not only bootstrapping authentication of each node, *e.g.*, by establishing an authenticated link between a node's attribute and a public key, but also to avoid the delivery of two, or more, certificates that link different identifiers to the same device.

Location-Limited Channels (LLCs), out-of-band channels to bootstrap authentication in wireless networks, were first introduced in (Stajano and Anderson, 1999) and successively inherited by (Balfanz et al., 2002). In the former work, the *Resurrecting Duckling* protocol sets up a relationship between two devices, by exchanging a secret key over an LLC established through *physical contact*. In the latter, a pre-authentication phase has been considered, where mobile hosts exchange data that will then be used for subsequent authentication of the parties at stake, *e.g.*, the parties may commit to their public keys over LLCs. LLCs must support: i) *demonstrative identification, i.e.*, identification based on physical context (*e.g.*, operators must be able to visibly control which devices are communicating with each other during a transmission); ii) *authenticity, i.e.*, it is not feasible, at least with high probability, that a host transmits over these channels without being detected.

We provide the network with an infrastructure of authorities and LLCs s.t. a certificate is delivered over the LLC to an user that has requested for it over the LLC. By exploiting LLC features like physical contact, (Stajano and Anderson, 1999), – or demonstrative identification and authenticity, (Balfanz et al., 2002) – a device is able to obtain a certificate only upon communicating under, a visible monitoring, a univocal credential (hereafter, UC). Such a credential could be either the serial number or the MAC address physically assigned to that device. Although a MAC address can be forged at software level, here we require its physical acquisition. The released certificate associates a public key (acquired by the authority together with UC) with the hash value of UC and it is signed by the private key of the authority.

By virtue of the media over which data are sent, a credential can be achieved by an authority in an unforgeable way. Thus, we extend the use of LLCs, originally introduced for pre-authentication between devices that successively communicate with each other. We propose them to assign unique certificates to mobile devices, thus precluding a device from re-certifying with a new UC.

One may comment that in the environment under investigation an approach based on a Certification Authority is not adequate, given the fully distributed and self-organizing topology of mobile ad hoc networks. Note that assumptions relying on central facilities intrinsically exist in the literature. As an example, *nuglets* in (Buchegger and Boudec, 2002) are universally known as valid by the community, and a tamper-proof module is required at each node. This makes it reasonable to think about an initial bootstrapping of the required infrastructure. Further, note that the use of the authority is here limited to an initial phase, in which bootstrapping of some required features is achieved, while other schemes, *e.g.*, (Lamparter et al., 2003; Zhong et al., 2003), rely on a central facility for the whole network lifecycle.

Identities in digital certificates. Although in standard X.509 based PKIs the same CA does not knowingly issue certificates to different entities under the same distinguished name, *i.e.*, all the valid certificates related to the same distinguished name are bound to the same entity, it could be possible for one entity to obtain several certificates, whose validity periods possibly overlap, validating different public keys under different pseudonyms. However, in many applications, there is the need for a third party to identify certificates that are bound to the same entity. Uncertainties in taking a decision are, for example, when: i) different pseudonyms are used by the same entity; ii) certificates issued by different CAs present a coincidence of names in the distinguished name field. Steps towards a possible solution are in (Pinkas and Gindin, 2004) (released on January, 2004, it will expire on July, 2004), where the concept of *Permanent Identifier* PI has been defined. PI is assigned to an entity by an Assigner Authority, and any certificate including the same identifier refers to the same entity, whatever the distinguished name may be. Since organizations can create links between different certificates through PIs, privacy problems can arise. Privacy issues are actually taken into account in (Jong-Wook and Polk, 2003) (released on October, 2003, it has expired on April, 2004; as declared by the authors, it should be considered as a work in progress — the same holds for (Pinkas and Gindin, 2004)), where the authors propose the notion of *Protected Identification Information, i.e.,* a commitment to PI.

Here, we consider a scenario with similarities to the above-depicted one. In particular, in our scenario there should be, at any time, only one valid certificate related to a certain device. (Actually, one may consider key rollover features for the renewal of certificates. We do not deal with key rollover in the current work. We assume here the lifetime of the network under consideration to be shorter than the lifetime of all the certificates at stake.)

Thus, we do not only need to find a credential *cr* peculiar to device *dv*, thus allowing a univocal association *cr–dv*, but also we need to assure that *dv*, at any time, does not possess more than one valid UC. We remark that this requirement is due since devices may easily use pseudonyms and IP addresses may be assigned by any mechanism (*e.g.,* through DHCP for dynamic assignment).

We rely on some physical attribute of the devices at stake (*e.g.,* the serial number), unforgeable since acquired by the authority through LLCs. At any time, the authority is responsible for the existence of one valid certificate related to a certain UC. Note that UCs do not appear in the certificate as a plaintext, hence our scheme may preserve privacy of the device at stake.

Finally, some words about open issues we do not deal with in this paper. Digital certificates have a validity period, after that they expire. One may also explicitly ask the certificate issuer to revoke the certificate, when, for instance, the user's private key is lost or compromised. In our framework, in case the private key is stolen, the responsibility for the thief's actions could fall on the original user. Indeed, the certificate binds the public key corresponding to the stolen private key to a UC that unequivocally identifies the original user. Dealing with expiration and explicit requests for revocation is actually a part of our ongoing research.

Adversary model. According to (Michiardi and Molva, 2003), "a selfish node does not directly intend to damage other nodes [...] by disrupting routing information [...] but it simply does not cooperate to the basic network functioning". Routing disruption attacks are those attacks, (Hu et al., 2002), where an adversary can route packets in a dysfunctional way, *e.g.,* it may attempt to make a suboptimal route to be chosen, for example a longer one. On the other hand, some authors assume that a selfish node may have also an active, malicious behavior, located somewhere between a non-cooperative behavior and a misbehavior aiming at damaging the others. As an example, in a reputation system providing awards, like the one in (Buttyan and Hubaux, 2002), a tamper-proof module is required at each node to prevent the node itself from intentionally increasing its nuglet counter.

The above considerations lead us to assume the following: a selfish node could not cooperate to the basic network functioning (*e.g.*, packet forwarding) and it could also illegally act in order to obtain benefits for sending its own packets. Suppose an adversary adds virtual nodes to a route which it belongs to, and suppose furthermore the adversary *owns* those added virtual nodes, then, it could consequently take the credit for the correct behavior of these nodes. Hence, it appears necessary to supply the network with protocols guaranteeing authenticity and integrity of control routing packets. To this aim, one may assume at routing level a secure version of DSR, like Ariadne (Hu et al., 2002). In our architecture all nodes have their cryptographic pair of keys certified before entering the network. Thus, we assume a secure Route Discovery based on Ariadne in its digital signature version (*i.e.*, Route Request is composed by nested signatures of IP addresses). We further assume the following extension: node *i*, receiving and processing a Route Request, appends its digital certificate $Cert^i$ to the request. Thus, Route Reply back to the source contains a certificate list, and each node receiving Route Reply is required to cache the list. Finally, node *i*, taking part in Route Discovery, appends to the request the hash of its univocal identifier, $h(UC_i)$.

Route Requests presenting more than one signature verifiable with the same public key should be marked as invalid requests and discarded. Further, there must be a correspondence between each fingerprint $h(UC_i)$ certified in $Cert^i$ and the one in the signed request.

A final remark: a certificate is validated by verifying its digital signature through the public key of the authority that has released the certificate. We assume that such public key is transmitted through the LLC.

5. The Credit Table

A table called the *credit table* (CT) is maintained at each node's side. Rows in the table consist of triples (*h(UC)*, *# debs*, *# creds*), where $h(UC)$ is the hash value of its univocal identifier, and *# creds* and *# debs* are the current values of the credits and debits counter related to that node.

The entity who maintains the table, say node *A*, quantifies the good behavior of the node corresponding to *h(UC)*, say node *B*, w.r.t. *B*'s past attitude to forward packets for *A*. From a complementary point of view, node *B*, that maintains in its turn memory about its behavior w.r.t. *A*, quantifies how much *A* can be indebted to *B*, *i.e.*, until when *B* can run the risk to forward packets for *A*. To limit the damages to forward packets for selfish nodes that do not return the favor, we give an upper bound over which it is not possible to help a node. We set this value to a default value $gap > 0$, equal for all nodes when they enter the network. Potentially, the entity who maintains a table can assign different values for *gap* to different entries in its table. For example, after deployment, a node A can set n different values $gap_1, \ldots, gap_n$, according to the perception A has about $node_1, \ldots, node_n$'s behavior (the latter being entries in A's table). Provided that node B behaves correctly, it forwards packets for node A if *creds* – *debs* $\leq$ *gap*, where *debs* and *creds* are the value of the debits/credits counters related to A in B's table.

B spends the earned credit *creds* at A's side when it starts sending packets along a route including A. From another point of view, suppose B needs to send packets to destination D: it either can recover an established path from its route cache or starts DSR Route Discovery, possibly returning several paths. In any case, by maintaining history of the past behavior of the network, B could choose the more *convenient* route (in terms of the nodes belonging to the route) rather than the shortest one.

CT Initialization. S asks for Route Discovery the first time it needs to send packets to destination D. The hash values of the univocal identifiers of the nodes in the returned path are the first entries in S's table. For all entries, the initial value assigned to both *debs* and *creds* is zero.

A CT table is initialized also by nodes belonging to a discovered path. Hence, a node forwarding a Route Reply message back to the source initializes its table by inserting those nodes listed in the path list, source included.

CT Maintenance. CT Maintenance is the mechanism by which the nodes update their CT. There could be two cases: 1) the node is the packet source S. In this case, updating the table happens once an authenticated proof has been received which testifies that the packet has been delivered to its destination. The authenticated proof is contained in a network-layer acknowledgment. When the source receives the acknowledgment, it increases by one the *debs* field correspondent to the identifiers of all the nodes constituting the current route. Thus, *debs* gives a measure, at S's side, of how many packets a certain node has forwarded for S. 2) The node belongs to the route from source S to destination D. Upon forwarding a packet, the node increases by one the *creds* field for S. Thus, the *creds* counter maintains information about how many packets the node has forwarded for S. Before forwarding a packet, the node checks if the difference of *creds* and *debs* related to S is greater (or equal) than the default value *gap* (or the value gap_S that the node has assigned to S). If so, the node forwards the packet for S (unless the node is a selfish one), otherwise it drops the packet.

We do not need a tamper-proof security module at each node, because the information recorded in a node's CT does not influence the rest of the network.

5.1 Authentication of data packets.

In the following, we consider a simple path from node S to node D through intermediate nodes A, B and C. We call route S-A-B-C-D Route 1.

In (Salem et al., 2003), the authors consider two kind of attacks to which a mobile ad hoc environment is prone. The first attack is when an intermediate node, say A, exploits a sub-route of Route 1, *e.g.*, A-B-C, to send its own packets. A may claim that those packets come from S and intermediate nodes B and C will charge S upon forwarding the packets. The second attack is the free riding attack. A may append (or substitute) its own payloads to the data packets transmitted from S to D over Route 1. The forged payloads may be consumed by C, that colludes with A, and B will charge S[2].

The above-mentioned attacks may be solved by exploiting part of a mechanism originally developed to sign digital streams, (Gennaro and Rohatgi, 2001). Let us suppose that S is to send blocks of *n* data packets to D. The construction exploits the technique of embedding the hash of the following packet in the current packet. Bootstrapping authentication is obtained by applying an initial digital signature, in combination with hash chaining. p_i be the *i*-th data packet sent by S (packet header plus meaningful payload). Then, the high-level formalization is as follows (we omit to explicitly denote the intermediate nodes A, B, C):

$$\begin{array}{llllll} 0) & S & \rightarrow & D & : p'_0 : & \{h(p'_1)\}_{pk_S^{-1}} \\ i) & S & \rightarrow & D & : p'_i : & p_i, h(p'_{i+1}) \quad i = 1, \ldots, n-1 \\ n) & S & \rightarrow & D & : p'_n : & p_n \end{array}$$

By doing so, source authentication is provided. Then, an intermediate node neither can append, or substitute, meaningful payloads to the data in the packet, since it should be able to forge digital signatures and hash functions, nor can claim its own transmissions to come from S, since it does not know the private key pk_S^{-1} [3]. Finally, this technique is applied on the whole packet, thus preserving integrity both of the data and of the packet header. Note that that construction assures authenticity to the ACK request option too.

Structure of network-layer ACKs and ACKs requests. Instead of requiring one ACK for every single packet arrived to its destination, we expect one ACK for every single block of *n* packets. This reduces the communication overhead on the way back from D to S.

To this aim, we propose the ACK request option in the first data packet header to have the following structure:

ACK Req Opt: $<< type, len, id, SAddr, DAddr >>$

where *type* specifies this is an acknowledgment request option, *len* is the length of the option, *id* is the identifier of the packet to be acknowledged. We set *id* = *n*, *i.e.,* the number of packets for which an acknowledgment is required, starting from this data packet. *SAddr* is the address of the node requesting the acknowledgment[4] and must be set to $(IP_S, h(UC_S))$. *DAddr* is the address

[2] Note that possible solutions to these attacks are beyond the scope of secure ad hoc routing protocols like Ariadne. Indeed, they address the authenticity of routing control packets, but not the one of data packets.
[3] As is common in security protocol analysis, we assume that digital signatures and hash functions cannot be forged, and moreover that it is not possible for an adversary to guess secrets of other participants.
[4] Extension already implemented in routing protocols specification like (Johnson et al., 2001).

of the node that should acknowledge the reception of n packets and must be set to $(IP_D, h(UC_D))$[5]. An acknowledgment request option must be ignored by all the intermediate nodes and must be processed only by D. If D correctly receives all the n packets for which ACK has been requested, it processes the request by sending back to *SAddr* an authenticated acknowledgment, whose structure is the following:

$$\text{ACK Opt: } << \{type, len, id, SAddr, DAddr, path\}_{pk_D^{-1}} >>$$

where *type* specifies this is an acknowledgment option, *len* is the length of the option, *id* = n is the number of packets that are acknowledged as received. *SAddr* is the address of the node originating the acknowledgment, *i.e.*, $(IP_D, h(UC_D))$. *DAddr* is the address of the node to which the acknowledgment is to be delivered, *i.e.*, $(IP_S, h(UC_S))$. With respect to routing protocols specification such as (Johnson et al., 2001), we have added the extension *path*, *i.e.*, the sequence of addresses as in the DSR Source Route Option in the header of the received packets. Here, *path* $=(IP_A, h(UC_A)), (IP_B, h(UC_B)), (IP_C, h(UC_C))$.

Since ACKs have smaller size than control and data packets, we assume that the nodes co-operate in sending ACKs back to the source. Further, given that S does not update its CT until it receives the proof that the packets have been delivered to their destination, it appears reasonable to assume that the intermediate nodes, that have already forwarded the packets, will cooperate in forwarding back to S the ACK.

If a node stops forwarding packets within a block, D never acquires the last packet, and it does not send back the ACK. As a consequence, the well-behaved nodes are never awarded by S. Possible patches to this drawback are: i) when D stops receiving packets, it notifies the anomaly to S, by sending an alert message (possibly over multiple routes); ii) an upper bound gap to the block size may be fixed. Thus, the intermediate nodes will not forward more than gap packets (*i.e.*, the limit we gave at the beginning of this section).

Like control and data packets, ACKs may be lost because of link breaks. We deal with this matter as follows: provided that D has in its Route Cache multiple routes to S, ACKs can be sent over all the available routes to S. Again, we assume to send (and forward) ACKs to be less power consuming than sending (and forwarding) data packets.

5.2 On the transfer of credits.

In the model we have presented, credits gained by node X can be *spent* only with the node for which X has forwarded something, say A. Intuitively, systems based on such rules can get stuck. We consider an established route

[5] The last field can be included as additional data in the Acknowledgment Request option in routing protocols specification like (Johnson et al., 2001).

A-B-X-C-D from source A to destination D. We know that if an intermediate node, say X, behaves correctly in forwarding packets for A, then it can reasonably rely on A for subsequent transmissions originated by X, when these transmissions involve routes including A. On the other hand, suppose X starts sending packets through route X-Y-W-Z and furthermore suppose that condition $creds - debs \leq gap_C$ is not fulfilled either for Y, or W, or both of them. In this case, X appears unable to correctly deliver packets to its destination.

Thus, we propose the notion of credits *transferring*, according to which X may rely on nodes not involved in the current route, say R, but likely willing to forward packets for X, for transfer their credits to the nodes in R. In particular, X will collect information regarding the nodes, in R, that are debtors to the nodes in X table for which it holds $creds - debs \leq 0$, meaning that X has made more favors to those nodes than they have made to X.

Credit transferring protocol. We consider three entities, namely: X, willing to send n packets over route R; A $\notin$ R, that is a X *first-hand debtor* (*i.e.,* A might accept to forward a certain number of packets for X); B $\in$ R, that is a X *second-hand debtor* (*i.e.,* B might accept to forward a certain number of packets for A).

$$\begin{array}{lcclll}
1) & X & \rightarrow & B & : \{\text{X fhd list}, X, B, n, nonce_X\}_{pk_X^{-1}} & \\
2) & B & \rightarrow & X & : \{A, B, X, m, nonce_X\}_{pk_B^{-1}} & m \leq n \\
3) & X & \rightarrow & A & : \{\{A, B, X, m, nonce_X\}_{pk_B^{-1}}\}_{pk_X^{-1}} & \\
4) & A & \rightarrow & X & : \{nonce_X, yes\}_{pk_A^{-1}} & \\
5) & X & \rightarrow & B & : \{\{nonce_X, yes\}_{pk_A^{-1}}\}_{pk_X^{-1}} &
\end{array}$$

The messages are signed by the private key of the sender. We assume that a special tag is contained in each message, to determine the message's step in the protocol. In message 1, X asks B if it agrees to accept a transfer of n credits from a node belonging to the list of X first-hand debtors. In particular, B should indicate who, among the nodes in the list, is its creditor. To maintain, in part, the user's privacy, items in the list should be hash values of the X first-hand debtors. The recipient of message 1 first computes the hash of the identifiers of the nodes to which it owes something, then it compares the hashes. Clearly, all the nodes processing message 1 may perform the same test, but, if they do not know those identifiers, X first-hand debtors remain unknown.

B indicates node A as a possible candidate for credit transferring (message 2). Through message 3, X asks A to transfer $m \leq n$ debits to B. A notifies the request acceptance to B (message 4). X notifies the credit transferring acceptance by forwarding message 4.

Provided that the protocol's participants behave correctly, they update their CT tables upon the reception of the messages. Updating CT tables is as reported in the scheme. In particular, X should refresh its table, as shown in Table # 1, upon receiving message 4 (meaning: X will not consider A as its debtor anymore, at least as far as m packets are concerned). In its turn, A refreshes its table, as shown in Table # 2, upon receiving a receipt that message 4 has been delivered to destination X. We rely on ACKs to convey receipts.

Upon reception of message 5, B will update its table as shown in Table # 3 (this is how we formalize the credit transferring operation). What appears after the updating is that in the past A has forwarded for B less packets than the packets A actually has forwarded, while X has forwarded for B more packets than the forwarded ones.

When m packets have finally reached their destination, not only X expects an acknowledgement, but also A. If A receives this proof, it updates its table as shown in Table # 4, meaning that A will not consider B as its debtor anymore.

Tables # 5 and # 6 show X and B updates, respectively, upon transmission of m packets. These standard updates follow the rules listed in Section 5.

A credit transferring protocol should be invoked in particular situations, *e.g.*, if the network under investigation supports high levels of mobility. We thus assume that the nodes cooperate in forwarding protocol messages and ACKs tied to a credit transferring, in order to maintain the basic network functioning, whereas the network could get stuck.

6. Conclusions

We have proposed to manage information about the forwarding behavior of the nodes in mobile ad hoc networks. Cryptography makes the information deduced by each node more reliable. Furthermore, we have proposed a mechanism to transfer the knowledge between different nodes. The novelty, w.r.t. previous mechanisms, is the avoidance of a central authority, the special stress on secure communication as well as on mechanisms to avoid that a user drops its identity. We are currently working on simulating the network performance to validate our approach. As future work, we plan to investigate new forms of

	debs	*creds*
id_A	-	creds-m
id_B	-	creds+m
other IDs	-	-

X table # 1

	debs	*creds*
id_X	debs-m	-
other IDs	-	-

A table # 2

	debs	*creds*
id_A	debs-m	-
id_X	debs+m	-
other IDs	-	-

B table # 3

Credit transferring

	debs	*creds*
id_B	-	creds-m
other IDs	-	-

A table # 4

	debs	*creds*
id_B	debs+m	-
other IDs	-	-

X table # 5

	debs	*creds*
id_X	-	creds + m
other IDs	-	-

B table # 6

credit transferring and routing protocols, based on the information gathered by users, as well as certificate expiration and key rollover issues.

References

Balfanz, D., Smetters, D., Stewart, P., and Wong, H. C. (2002). Talking to Strangers: Authentication in Ad-Hoc Wireless Networks. In *Proc. of NDSS'02*. The Internet Society.

Buchegger, S. and Boudec, J. L. (2002). Performance Analysis of the CONFIDANT Protocol. Cooperation Of Nodes – Fairness In Dynamic Ad-hoc Networks. In *Proc. of ACM MobiHoc'02*.

Buttyan, L. and Hubaux, J. (2002). Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. *ACM/Kluwer Mobile Networks and Applications (MONET)*, 8(5).

Gennaro, R. and Rohatgi, P. (2001). How to Sign Digital Streams. *Information and Computation*, 165(1):100–116.

Hu, Y., Perrig, A., and Johnson, D. (2002). Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth ACM International Conference on Mobile Computing and Networking (Mobicom 2002)*.

Johnson, D., Maltz, D., and Broch, J. (Addison-Wesley, 2001). DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks. *Ad Hoc Networking, chapter 5*, pages 139–172.

Jong-Wook, P. and Polk, T. (Internet Draft, October 2003). Internet X.509 Public Key Infrastructure Subject Identification Method SIM.

Lamparter, B., Plaggemeier, M., and Westhoff, D. (2003). About the Impact of Co-operation Approaches for Ad Hoc Networks. In *Proc. of ACM MobiHoc'03*.

Marti, S., Giuli, T., Lai, K., and Baker, M. (2000). Mitigating Routing Misbehaviour in Mobile Ad Hoc Networks. In *Proc. of MobiCom'00*, pages 255–265. ACM.

Michiardi, P. and Molva, R. (2002a). Core: A Collaborative Reputation Mechanism to Enforce Node Cooperation in Mobile Ad Hoc Networks. In *Proc. of CMS'02*.

Michiardi, P. and Molva, R. (2002b). Simulation-based Analysis of Security Exposures in Mobile Ad Hoc Networks. In *Proc. of European Wireless'02*.

Michiardi, P. and Molva, R. (2003). A Game Theoretical Approach to Evaluate Cooperation Enforcement Mechanisms in Mobile Ad hoc Networks. In *Proc. of WiOpt'03*.

of the IEEE Computer Society, L. M. S. C. (1999). Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. In *IEEE Standard 802.11, 1999 Edition*.

Pinkas, D. and Gindin, T. (Internet Draft, January 2004). Internet X.509 Public Key Infrastructure Permanent Identifier.

Salem, N. B., Buttyan, L., Hubaux, J., and Jakobsson, M. (2003). A Charging and Rewarding Scheme for Packet Forwarding in Multi-hop Cellular Networks. In *Proc. of ACM MobiHoc'03*.

Stajano, F. and Anderson, R. (1999). The Resurrecting Duckling: Security Issues for Ad-Hoc Wireless Networks. In *Proc. of 7th Security Protocols Workshop*, volume LNCS 1796, pages 172–194.

Urpi, A., Bonuccelli, M., and Giordano, S. (2003). Modelling Cooperation in Mobile Ad-hoc Networks: a Formal Description of Selfishness. In *Proc. of WiOpt'03*.

Zhong, S., Chen, J., and Yang, Y. (2003). Sprite: a Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks. In *Proc. of IEEE Infocom'03*.

Zhou, L. and Haas, Z. (1999). Securing Ad Hoc Networks. *IEEE Network*, 13(6):24–30.