A Secure Incentive Architecture for Ad-hoc Networks

Qi He* Dapeng Wu† Pradeep Khosla‡

Abstract

In an ad-hoc network, intermediate nodes on a communication path are expected to forward packets of other nodes so that the mobile nodes can communicate beyond their wireless transmission range. However, because wireless mobile nodes are usually constrained by limited power and computation resources, a selfish node may be unwilling to spend its resources in forwarding packets which are not of its direct interest, even though it expects other nodes to forward its packets to the destination. It has been shown that the presence of such selfish nodes degrades the overall performance of a non-cooperative ad hoc network.

To address this problem, we propose a Secure and Objective Reputation-based Incentive (SORI) architecture to encourage packet forwarding and discipline selfish behavior. Different from existing schemes, under our architecture, the reputation of a node is quantified by objective measures; the propagation of reputation is efficiently secured by a one-way-hash-chain-based authentication scheme; and secure routing is in place. Armed with the reputation-based mechanism, we design a punishment scheme to penalize selfish nodes. The experimental results show that the proposed scheme can successfully identify selfish nodes and punish them accordingly.

*Carnegie Mellon University, Dept. of Electrical & Computer Engineering, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA. Email: qihe@cmu.edu.

†Please direct all correspondence to Prof. Dapeng Wu, University of Florida, Dept. of Electrical & Computer Engineering, P.O.Box 116130, Gainesville, FL 32611, USA. Tel. (352) 392-4954, Fax (352) 392-0044, Email: wu@ece.ufl.edu. URL: http://www.wu.ece.ufl.edu.

‡Carnegie Mellon University, Dept. of Electrical & Computer Engineering, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA. Tel. (412) 268-5090, Fax (412) 268-5787, Email: pkk@ece.cmu.edu. URL: http://www.ece.cmu.edu/about/deptheadbio.shtml.

Key Words: security techniques and systems, non-cooperative ad hoc networks, packet forwarding, selfish behavior, incentive, reputation.

1 Introduction

The convergence of wireless communication (*e.g.*, IEEE 802.11 and Bluetooth) and mobile computing devices (*e.g.*, laptop, personal digital assistant (PDA), and wearable computer) offers great promise of providing us with unprecedented connectivity and mobility which enable us to enjoy untethered computing, at any place and at any time. One of the attractive paradigms under such a convergence is ad-hoc networks which can be easily and dynamically formed by a group of wireless mobile nodes without assistance from any fixed communication infrastructure such as base stations or access points. In an ad-hoc network, the transmission range of a mobile node is limited due to the power constraint, and there is no fixed communication infrastructure to facilitate packet forwarding; hence, the communication between two nodes beyond the transmission range relies on intermediate nodes to forward the packets. However, because mobile nodes are typically constrained by power and computing resources, a selfish node[1] may not be willing to use its computing and energy resources to forward packets that are not directly beneficial to it, even though it expects others to forward packets on its behalf. It has been shown [8] that the presence of such selfish nodes significantly degrades the overall performance of a non-cooperative ad-hoc network. Here, by "non-cooperative", we mean that a node is not willing to forward packets of other nodes unless it can benefit from the packet forwarding.

To address the above problem, we propose a Secure and Objective Reputation-based Incentive (SORI) architecture to encourage packet forwarding and discipline selfish nodes. The unique features of our architecture are that 1) the reputation of a node is quantified by *objective* measures, 2) the propagation of reputation is *computationally-efficiently secured* by a one-way-hash-chain-based authentication scheme, 3) the reputation of a node is only propagated to its neighbors but not entire network since the reputation of a node is only used by its neighbors in our scheme, which *reduces communication overhead*, and 4) routing is effectively secured. Equipped with the reputation-based mechanism, we design a punishment scheme to penalize selfish nodes. The experimental results show that the proposed scheme

[1] A node that does not fulfill the responsibility of forwarding packets is called "selfish node" or "misbehaving node". "Selfish node" and "misbehaving node" are two interchangeable terms in this paper.

can successfully identify selfish nodes and punish them accordingly.

The incentive schemes for packet forwarding in the literature basically fall into two categories, namely, *reputation-based* schemes and *pricing-based* schemes.

Reputation-based schemes utilize reputation in routing [8] and/or enforcing punishment [1]. However, the existing reputation-based schemes suffer from lack of effective mechanisms to measure and propagate reputation. Without quantitative and objective ways to measure reputation, and secure ways to propagate reputation, a reputation-based incentive scheme would not serve the purpose of stimulating packet forwarding, since reputation can be easily manipulated by selfish nodes in this case. Hence, this paper proposes a quantity to objectively measure reputation of a node, and a secure mechanism to propagate reputation, with the aim of resolving the drawbacks of the existing reputation-based incentive schemes.

Pricing-based schemes [2, 5, 16] treat packet forwarding as a service that can be priced, and introduce some form of virtual currency to regulate the packet-forwarding relationships among different nodes. However, these schemes require either tamper-resistant hardware[2] [2] or virtual banks (trust authorities) that all parties can trust [5, 16]. In the case where tamper-resistant hardware is used, if a node (say, node A) sends much less traffic than another node (say, node B), node A may drop most of the packets of node B without losing anything, since node A need not earn more money than necessary to support the small volume of its own traffic. In the case where a trust authority or virtual bank is required, it needs assistance from a fixed communication infrastructure to implement the incentive schemes, which is not applicable for a pure ad hoc network. In contrast to these, our SORI architecture is based on reputation and hence does not require tamper-resistant hardware or trust authorities; in addition, under our scheme, each node are motivated to maintain a good reputation in order to keep a desirable quality of its network connectivity.

The remainder of the paper is organized as follows. In Section 2, we specify assumptions made in this paper. Under these assumptions, we first develop our basic scheme in Section 3

[2]A tamper-resistant hardware contains a counter indicating the amount of available virtual currency; it increases the counter when the node forwards packets, and decreases the counter when the node sends out its own packet; it can be accessed only by its intended user(s) and only for its intended purposes.

and then improve the basic scheme with security enhancements described in Section 4. In Section 5, we show simulation results to demonstrate the effectiveness of our scheme. Section 6 discusses the related work. In Section 7, we conclude the paper.

2 Assumptions

In this paper, we make the following assumptions.

1. The nodes in an ad hoc network under our consideration are *non-cooperative* in packet forwarding, that is, a node is not willing to forward packets of other nodes unless it can benefit from the packet forwarding. If nodes are cooperative, *e.g.*, in military ad hoc networks, there is no need to use incentive mechanisms.

2. There is *no conspiracy* among nodes.

3. *Broadcast transmission*: A packet can be received by all the neighbors of the transmitting node (within its transmission range) because of the broadcast nature of the wireless medium.

4. *Desire to communicate*: All the participating nodes have the desire to communicate with some others.

5. *Invariant identity*: No node changes its identity during its life time. This assumption is reasonable since changing identity of a node results in unreachability of packets destined to the node, which violates the previous assumption "desire to communicate".

6. *Selfish but not malicious*: A node may be selfish in terms of conservation of power and computing resources, but not malicious, which means that it will not try something that could be more expensive in consuming energy and computing resources than cooperating in packet forwarding. In other words, a node will not do something that cannot conserve its computation/energy resource. For example, a node will not sabotage the network by denial-of-service attacks, which consumes a great deal of its energy.

7. *Promiscuous mode*: Each node operates in a promiscuous mode, *i.e.*, each node listens to every packet transmitted by its neighbors even if the packet is not intended for the node; and each node is able to determine who transmits the packet. The promiscuous mode is (typically) required for ad hoc networks unless deterministic medium access control such as time division multiple access is employed; without listening to every packet transmitted by its neighbors, a node may miss some packets destined to it.

3 Basic Scheme

This section presents the basic scheme of our incentive mechanism, under the assumptions listed in Section 2. The basic scheme consists of three components, namely, neighbor monitoring, reputation propagation, and punishment, which are described as follows.

3.1 Neighbor Monitoring

In our scheme, neighbor monitoring is used to collect information about the packet-forwarding behavior of the neighbors. Due to the promiscuous mode that we assume, a node is capable of overhearing the transmissions of its neighbors. With this capability, a mobile node N can maintain a *neighbor node list* (denoted by NNL_N) which contains all of its neighbor nodes that node N learns of by overhearing. In addition, node N keeps track of two numbers, for each of its neighbor (denoted by X), as below.

- $RF_N(X)$ (Request-for-Forwarding): the total number of packets that node N has transmitted to X for forwarding.
- $HF_N(X)$ (Has-Forwarded): the total number of packets that have been forwarded by X and noticed by N.

The two numbers are updated by the following rules. When node N sends a packet to node X for forwarding, the counter $RF_N(X)$ is increased by one. Then N listens to the wireless channel and check whether node X forwards the packet as expected. If N detects that X

has forwarded the packet before a preset time-out expires, the counter $HF_N(X)$ is increased by one.

Given $RF_N(X)$ and $HF_N(X)$, node N can create a record called *local evaluation record* (denoted by $LER_N(X)$), for the neighbor node X. The record $LER_N(X)$ consists of two entries, that is, $G_N(X)$ and $C_N(X)$, where $G_N(X) = \frac{HF_N(X)}{RF_N(X)}$ and $C_N(X)$ is a metric called *confidence*, used to describe how confident node N is for its judgement on the reputation of node X. In our scheme, we set $C_N(X) = RF_N(X)$; that is, the more packets transmitted to X for forwarding, the better estimation about how well the neighbor X does forwarding.

3.2 Reputation Propagation

With the fore-mentioned neighbor monitoring, a node could build a record of the reputation of its neighboring nodes. However, our initial experimental result shows that actions (dropping selfish nodes' packets) solely based on one's own observation of its neighbors cannot effectively punish selfish nodes. To address this problem, reputation propagation is employed to have neighbors share the reputation information of other nodes, so that a selfish node will be punished by all of its neighbors (who share the reputation information about its misbehavior) instead of just the ones who get hurt by the selfish node. The reputation propagation works as follows:

1. Each node N periodically updates its $LER_N(X)$ for each neighbor node X based on the changes of $RF_N(X)$ and $HF_N(X)$, and it broadcasts the updated record to its neighborhood if $G_N(X)$ has been significantly changed.

2. Node N uses its $LER_N(X)$ and $LER_i(X)$ (where i is in the NNL_N) to calculate its *overall evaluation record* of X (denoted by $OER_N(X)$) as below:

$$OER_N(X) = \frac{\sum_{i \in NNL_N \cup \{N\}, i \neq X} \lambda_N(i) \cdot C_i(X) \cdot G_i(X)}{\sum_{k \in NNL_N \cup \{N\}, k \neq X} \lambda_N(k) \cdot C_k(X)} \quad (1)$$

   where $\lambda_N(i)$ is the *credibility* that node i has earned from the perspective of node N. In current scheme, we choose $\lambda_N(i) = G_N(i)$, especially, $\lambda_N(N) = 1$, and $\lambda_N(i) = 0$ if

$RF_N(i) = 0$, which means a node would not have any credit from N if it has not been requested by N to forward any packet.

3.3 Punishment

With the reputation measure $OER_N(X)$ obtained, node N can punish its neighbor X by probabilistic dropping as follows. If the $OER_N(X)$ is lower than a preset threshold, node N takes punishment action by probabilistically dropping the packets originated from X. The probability of dropping is

$$p = \begin{cases} q - \delta & \text{if } q > \delta \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

where $q = 1 - OER_N(X)$ and $0 < \delta < 1$ is the margin introduced for the following consideration: a dropping action could be occasionally triggered by some phenomena such as collision, rather than selfishness. Without the margin, two nodes may keep increasing the dropping probability and consequently fall into a retaliation situation. The margin is designed to help well-behaving nodes avoid this situation by treating each other a little bit more generously.

To summarize, this section presents our basic incentive scheme, which consists of neighbor monitoring, reputation propagation, and punishment. Neighbor monitoring is to collect information about misbehavior of neighbors and objectively quantify reputation of neighbors. Reputation propagation is aimed at sharing information among neighboring nodes to make the reputation measure more accurate (from statistical perspective, *i.e.*, the weighted averaging in (1) helps removing dependence of the reputation measure on location and node identity). Punishment is to encourage packet forwarding and discipline selfish nodes.

The basic scheme is vulnerable to some tricks played by selfish nodes and we improve it by security enhancements presented in section 4.

Remark 1 *Objective quantification of reputation*

- *The reputation of a node is objectively measured based on the packet forwarding ratio of the node, and it directly affects the service the node can obtain from other nodes.*

- *The reputation of a node is weighted by the confidence, which is objectively determined by the number of packets that the node is requested to forward, and the total number of packets forwarded and observed. Such a weighting is rational from a statistical perspective, since the more samples that are used, the more reliable the estimation of reputation will be.*

- *The reputation of a node is also weighted by the credibility of the nodes which contribute to the calculation of the reputation.This makes it difficult for a selfish node to play with multiple identities, trying to use one identity to propagate fake information in order to improve its reputation under another identity. Since as a selfish node who intrinsically has very little credibility under any of its identities (otherwise it wouldn't be a selfish node), weighting the reputation by credibility effectively limits the contribution that one identity can make to the calculation of the reputation under another identity. That is, if an identity (denoted by I_A) is used by a selfish node, this selfish node cannot effectively get a good reputation for the identity I_A by using another identity to propagate fake information. An extreme example is that a selfish node which never helps others on packet forwarding will earn a zero credibility from any other node and therefore cannot make any contribution to the calculation of the reputation under any other identity.*

- *Reputation propagation is to share information among neighboring nodes to make the reputation measure more accurate. For example, node X drops all the packets from node N but forwards all the packets from all the other nodes; hence, node N has $LER_N(X) = 0$ and all the other nodes i ($i \neq N$) have $LER_i(X) = 1$. Due to reputation propagation in our scheme, node X will be punished by all its neighbors; without reputation propagation, node X will be punished by node N only. Therefore, reputation propagation can help discourage selfish behavior.*

- *Since we assume there is no conspiracy among nodes, a well-behaving node will not help to falsify good reputation for a misbehaving node; for example, a node i with good reputation will not propagate fake $LER_i(X)$ to boost the reputation of another node X, which has a bad reputation.*

- *Since we assume a node may be selfish but not malicious, a node with good reputation will not spread fake bad-reputation report about its neighbors, which may result in punishment on these neighbors; another reason for a node not to do so, is that spreading fake reputation report will not conserve its computation/energy resource.*

- *A limitation of the reputation calculation is that the objectivity of the calculated reputation depends on the probability of transmission collision. This is because the measure $HF_N(X)$ may not reflect the total number of packets that are transmitted from node N to node X and have been correctly received by node X, due to packet collision in the wireless medium. However, our simulation results in Section 5 show that our incentive scheme can identify selfish nodes and punish them under a light or medium network load. For the case when the network is heavily loaded, congestion control is required. Obviously, no incentive scheme will work well under a heavy network load, without the aid of congestion control.*

4 Security Enhancement

This section presents security enhancements to fix the vulnerabilities of the basic scheme, which could be potentially exploited by selfish nodes.

A selfish node may play the following tricks in order to benefit itself without being detected by the basic scheme:

Problem 1 (P1): Impersonate a node that is near the selfish node and has a good reputation, in order to send its own packets (using other's identity).

Problem 2 (P2): Impersonate a node with a good reputation, in order to broadcast fake observation information in order to boost its reputation calculated by other nodes.

Problem 3 (P3): Manipulate the routing information during routing path discovery in order to have the data packets get around the selfish node itself so that the selfish node can avoid the responsibility for forwarding other's data packets without being detected. Here, we assume the dynamic source routing (DSR) protocol [6] is used.

To address the first two problems, we propose an identification and authentication method based on a one-way-hash chain, which is presented in Section 4.1. To mitigate the third problem, we design a secure dynamic routing protocol, which is described in Section 4.2.

4.1 One-way-hash Chain for Identification and Authentication

To address the problems P1 and P2, a natural solution is to use identification and authentication since with identification and authentication, well-behaving nodes can be identified (and rewarded) and misbehaving nodes can also be identified (and punished). To design an effective identification and authentication scheme, one needs to consider the unique property of ad hoc networks, which is different from wired networks and wireless cellular networks. This property of ad hoc networks is *no fixed infrastructure*. Due to this feature, our design for identification and authentication does not rely on any infrastructure to help the communication or management of mobile nodes. Note that this design principle is different from those assuming the existence of an authentication infrastructure such as a public key infrastructure (PKI) or a "friendship" infrastructure as in [1].

Based on the above design principle, we propose an authentication mechanism based on a one-way-hash chain [7] as below. Node N gets its identity, denoted by ID_N, by choosing a random number r_N and recursively applying a pseudo-random function h on r_N by k times,[3] that is, $ID_N = H_k(r_N)$ which is recursively obtained by

$$H_i(r_N) = \begin{cases} h(H_{i-1}(r_N)) & \text{if } i \in \{1, 2, \cdots, k\} \\ h(r_N) & \text{if } i = 0 \end{cases} \tag{3}$$

When N is joining an ad-hoc network, it broadcasts its identity ID_N and all its neighbors receive this identity and put this identity into their NNLs. The neighbors will use this identity to authenticate messages originated or forwarded by this node (identified by ID_N) thereafter.

[3]The k is the length of the one-way-hash chain and this length limits the maximum numbers of the messages to generate before a new one-way-hash chain must be created. How to handle unlimited message authentication by switching one-way-hash chains was discussed by Perrig *et al.* [13].

Next, we describe our procedure of message authentication. Our procedure of authentication is the same as that suggested by Perrig *et al.* [13] for broadcast authentication. Node N partitions the time into equal intervals and assigns the i-th interval with a key (denoted by K_i) which is $K_i = H_{k-i}(r_N)$ in the one-way-hash chain. The messages sent in an interval will be accompanied by a message authentication code (MAC) which is computed with the corresponding key K and the message M as the input, denoted by $MAC(K, M)$. For instance, the content of packet P_i sent in the i-th interval is

$\{M_i||MAC(K_i^{'}, M_i)||K_{i-d}\}$

where M_i is the message to be sent in the i-th interval; the key $K_i^{'}$ is obtained from $K_i^{'} = f(K_i)$ where f is the second pseudo-random function[4] and $K_i = H_{k-i}(r_N)$; d is the key disclosure delay (for example, in the $(i-d)$-th interval, the message is authenticated by key K_{i-d}, and key K_{i-d} will be disclosed in the i-th interval). Once receivers receive a packet, they check if the key used for the MAC is already disclosed. If the key has not been disclosed, they cache the message and will check its authenticity at the time when the K_i is disclosed; otherwise, they discard the packet because the key was disclosed before they received the packet and the MAC could be potentially forged. In addition, a packet with an invalid MAC will be discarded.

Note that a misbehaving node N will not generate a new random number $\hat{r}_N$ and the associated one-way-hash chain to get rid of its bad reputation (assuming node N generated a random number r_N previously for authentication). This is because the identity ID_N of node N is already set to $H_k(r_N)$ and changing ID_N to $H_k(\hat{r}_N)$ violates the assumption of invariant identity (see Section 2).

The key feature of our authentication scheme is that the one-way-hash chains are designed to 'identify' mobile nodes; specifically, the identity of a node N is associated with the keys that are used to 'authenticate' packets, *i.e.*, $ID_N = H_k(r_N)$. In contrast to this, most existing schemes separate the identification of mobile nodes from message authentication; and hence they require a PKI or other trust management systems to identify mobile nodes,

[4]The second pseudo-random function is used to avoid using the same key multiple times in different cryptographic operations, hash chains and MACs.

which are usually not practical for a non-cooperative ad-hoc network.

Our authentication scheme makes it difficult for a selfish node which has a bad reputation to send out its packets or broadcast fake observation information to affect the others calculation of its reputation by impersonating a node with a good reputation. This is because the MAC is computationally difficult to forge without the key of that node. Moreover, our design is efficient because the authentication is done with a one-way-hash function which is computationally much cheaper than the digital signature used in many other schemes. This is particularly desirable since most mobile devices are constrained by energy and computation resource.

Like any other security measure, our scheme also pays some cost. A cost is that our scheme requires time synchronization to make a time-slotted system. However, the synchronization can be loose and there are a number of simple ways to achieve synchronization as suggested by [13].

4.2 Secure Dynamic Routing Protocol

To mitigate the problem P3, we enhance the DSR routing protocol by a security measure. We first briefly describe the DSR protocol [6]. The DSR is an on-demand source routing protocol. The key component of the DSR is a route discovery protocol, which is invoked by a source node S when node S does not know any path from node S to the destination D. It works as below. First, node S initiates a route request packet $rreq$ and broadcasts it to its neighbors. Then, the neighbors in turn append their own address to packet $rreq$ and broadcast it. This process continues until packet $rreq$ reaches the destination D. Since packet $rreq$ contains a complete path from node S to node D when it reaches node D, node D can send a route request reply packet $rrep$ (containing the path information) back to node S through the reverse path. Once node S receives packet $rrep$, the path from node S to node D (contained in packet $rrep$) is confirmed. Then, the packets to node D can be forwarded through this identified path.

We start with discussion about the vulnerabilities of our basic scheme in Section 3. First,

we are not worried about that a selfish node may not be cooperative on forwarding route request packets since doing so will damage its reputation. What is of our concern is that a selfish node may play some tricks when forwarding route request *rreq* and route request reply *rrep*. For example, a selfish node may provide a source node with a fake path, which does not include the selfish node and hence looks shorter than the valid path that includes the selfish node. Since usually the shortest path would be picked, the shorter path not containing the selfish node is selected; hence, the selfish node avoids the responsibility of forwarding data packets and this cheating is not detected! This kind of misbehavior would severely degrade the performance of the network [10] since the packets transmitted along the invalid path are unlikely to reach the destination.

To fake a path not including a selfish node N, node N may take the following actions.

Case 1 (node N is a neighbor of the source S): First, node N participates in the normal procedure of the route discovery protocol; and assume a route request reply *rrep* conveys a valid path record (containing node N) to node S. Then, node N can fake a route request reply *rrep* containing an arbitrary path of length l that is shorter than the discovered path containing node N; denote this fake path by $\{N_0(=S), N_1, ..., N_l(=D)\}$, where D is the destination, and N_1 is any neighbor of S other than node N. As long as $N \notin \{N_1, N_2, ..., N_l\}$, the selfish node N will not be asked to forward the packets from node S to node D.

Case 2 (node N is not a neighbor of the source S): Node N also participates in the normal procedure of the route discovery protocol. However, upon receiving a route request *rreq* including a route record, say $\{N_0(=S), N_1, ..., N_{j-1}\}$, node N broadcasts two packets, denoted by $rreq_1$ and $rreq_2$, to the next hop; $rreq_1$ contains node N in its route record but $rreq_2$ does not. Then, the destination D will return two route request reply messages $rrep_1$ and $rrep_2$; $rrep_1$ contains a route record $\{N_0(=S), N_1, ..., N_{j-1}, N_j(=N), N_{j+1}, ..., N_l(=D)\}$ (where l is the number of hops on the route) and $rrep_2$ contains $\{N_0(=S), N_1, ..., N_{j-1}, N_{j+1}, ..., N_l(=D)\}$. Upon receiving $rrep_1$ and $rrep_2$, node N forwards both $rrep_1$ and $rrep_2$ to node N_{j-1}. Note that here we assume node N_{j+1} does not check whether node N is its neighbor and hence node

N_{j+1} will broadcast $rrep_2$ and node N will receive $rrep_2$ even though node N is not on the path as specified by $rrep_2$. In this way, source S obtains a (fake) shorter path that does not include node N. Therefore, the selfish node N will not be asked to forward the packets from node S to node D.

Alternatively, node N may fake an invalid path of the same length as that of the valid one, by replacing its own ID by a spoofed ID; in so doing, it may end up with a path like $\{N_0(= S), N_1, ..., N_{j-1}, N_j^{'}(\neq N), N_{j+1}, ..., N_l(= D)\}$. Hence, the probability that the valid path (containing node N) would be selected is reduced by 50%.

We address Case 1 by enhancing the DSR protocol with a source-destination authentication scheme. The key assumption of our scheme is that the source S and the destination D can verify the authenticity of the messages in their communications. This assumption is reasonable since the communication between S and D implies that the existence of an association between S and D and a secret key can be shared between them through the association. Accordingly, the communication partners S and D can use this secret key to compute MACs in order for them to authenticate the messages in their communications. In fact, because of its feasibility, this assumption is usually the basis of many secure routing protocols, *e.g.*, [12].

Our source-destination authentication scheme is as below. Upon receiving a route request $rreq$, the destination D computes a message authentication code, denoted by MAC_d, for the message $\{N_0(= S), N_1, ..., N_l(= D), seq, MAC_s\}$, where $\{N_0(= S), N_1, ..., N_l(= D)\}$ is the discovered path, seq is a query sequence number, and MAC_s is the MAC for $\{N_0(= S), seq\}$ and is generated by the source S; the query sequence number seq, used only by S for query identification, allows D to discard replayed queries. Then, the destination D appends MAC_d to its reply packet, $rrep = \{N_0(= S), N_1, ..., N_l(= D), seq, MAC_s, MAC_d\}$, and sends the $rrep$ to S in the reverse direction of the discovered path. A route reply $rrep$ will not be accepted by S unless the MAC_d contained is valid. Since the MAC is computationally infeasible to be forged without the secret key shared by S and D, it is difficult for a selfish node to have the source accept a fake path.

To address Case 2, we borrow the idea originated from the associativity-based routing

(ABR) [15]. Under the ABR protocol, every node periodically transmits beacons indicating its existence to its neighbors, and also receives beacons from its neighbors. Each node maintains a table, each entry of which registers the number of beacons the node has heard from one of its neighbors; different entry of the table corresponds to different neighbor. These numbers are used in the ABR to measure the node's connectivity relationship (a.k.a. associativity) with its neighbors. Under the ABR, the associativity measure is used to select the most reliable path. Different from the ABR, we utilize the associativity measure to identify 'inactive' neighbors, which could be faked by selfish nodes. A neighbor X of node N is regarded as being active if node N periodically receives the beacons from node X; otherwise, node X is regarded as being inactive. Because of transmission collision in the shared channel, there is no guarantee that a node N receives every transmitted beacon from its neighbors. So, a practical way of deciding being active is as below: node N marks a neighbor X as being active if the number of successfully received beacons from node X during a certain time period exceeds a pre-determined threshold.

Assuming a time-slotted system as in Section 4.1, with the above tag of being active or inactive, and the one-way-hash-chain based authentication scheme described in Section 4.1, we propose an enhanced route discovery protocol for the DSR as below.

1. The source node S initiates the route discovery protocol, by constructing and broadcasting a route request packet $rreq = \{S, D, seq, MAC_s\}$, where seq is the query sequence number and MAC_s is the MAC for message $\{S, D, seq\}$, computed with the key shared by S and D.

2. Once the j-th intermediate node X (along the path) receives the request $rreq$ at the i-th interval, node X works as below.

   (a) If the identity of node X is already contained in the route record of the $rreq$, discard the request.

   (b) If the node that forwarded this $rreq$ is not an active neighbor of X, discard the request.

(c) If the route record in the $rreq$ contains more than one active neighbor of X, discard the request.[5]

(d) If the MAC_{j-1} (described as below) in the $rreq$ is incorrect,[6] discard the request.

(e) Otherwise, node X removes MAC_{j-1} from the $rreq$; append its identity to the route record; compute a message authentication code MAC_j for $\{N_0(= S), ..., N_{j-1}, N_j(= X), seq, MAC_s\}$, using the key of the i-th interval; and broadcast the request $rreq = \{N_0(= S), ..., N_{j-1}, N_j(= X), seq, MAC_s, MAC_j, K_{i-d}\}$, where d is the key disclosure delay and K_{i-d} is the key of the $(i-d)$-th interval.

3. When the request $rreq$ reaches the destination D, node D works as below.

(a) If the seq in the $rreq$ is not new, discard the request.

(b) If the node that forwarded the $rreq$ is not an active neighbor of D, discard the request.

(c) If the route record in the $rreq$ contains more than one active neighbor of D, discard the request.

(d) If the MAC in the $rreq$ is incorrect, discard the request.

(e) Otherwise, node D constructs a route request reply $rrep = \{N_0(= S), N_1, ..., N_l(= D), seq, MAC_s, MAC_d\}$, where MAC_d is the MAC for the message $\{N_0(= S), N_1, ..., N_l(= D), seq, MAC_s\}$. Then, node D sends the $rrep$ to source S in the reverse direction of the discovered path.

4. When the $rrep$ reaches node S, node S operates as below.

(a) If the MAC_d in the $rrep$ is incorrect, discard the $rrep$.

(b) Otherwise, accept the path in the $rrep$ as a valid path, and pick the shortest path among all the valid paths to send data packets.

[5]This could help the source node find an optimal path. But this step is optional.

[6]To authenticate MAC_{j-1}, node X may need to wait for key disclosure from the upstream node N_{j-1}. To reduce this delay, immediate authentication can be achieved using the method presented in [13].

Under our secure routing protocol, it is unlikely for a selfish node N to play tricks to make the source node pick a fake (shorter) path. A possible trick is that node N periodically broadcasts the beacons on behalf of its neighbor, say node X, making node X appear to be the neighbor of node Y, which is a neighber of node N but is two hops away from node X. However, sending these fake beacons may cost more energy resource than forwarding others' packets.

In our scheme, beaconing is required and may incur quite a bit overhead; however, if an ABR-type of routing protocol is used, beaconing is readily available and we do not have to pay an extra cost. To reduce the energy consumed by beaconing, we can increase the beaconing interval; or a node does not need to send beacons unless it has not communicated for a certain time period, and a node N marks its neighbor X as being active if node N learns that there are certain amount of communication from node X during the time period.

5 Simulation Results

In this section, we implement our basic incentive scheme on a simulator and evaluate its performance under various settings. The purpose of this section is to demonstrate the effectiveness of our scheme in identifying selfish nodes and punish them accordingly. This section is organized as follows. Section 5.1 describes the simulation setting, while Section 5.2 illustrates the performance of our scheme.

5.1 Simulation Setting

Our incentive scheme is implemented on ns-2 [11]. We simulate a wireless ad-hoc network with 50 mobile nodes randomly deployed in an area of 670×670 square meters. We use the Distributed Coordination Function (DCF) of IEEE 802.11 as the medium access control layer protocol, and dynamic source routing (DSR) as the routing protocol. The radio transmission range for each node is 250 meters and the transmission data rate is 2 Mbits/s. The physical layer model is either the free space or the two-ray propagation model [14], depending on the separation distance between the transmit antenna and the receive antenna. The height of

both the transmit antenna and the receive antenna is 1.5 meters.

The random waypoint mobility model [4] is used to generate the moving direction, the speed and the pause duration of each node. The node speed is uniformly distributed between 0 and 20 m/s, and the pause duration is exponentially distributed with an expectation of 600 seconds.

Among the 50 nodes, 5 nodes are randomly selected and assigned as selfish nodes. A selfish node in our simulation probabilistically drops packets from other nodes unless it is the destination of the packet.

For each simulation, we first set the total number of connection, denoted by N_{conn}. Then, we randomly generate N_{conn} source-destination pairs; the generated source-destination pairs may be duplicated, that is, the same source-destination pair may have multiple connections. Each connection lasts for 10 simulated seconds. Once a connection is terminated at the end of the 10th simulated second, a new source-destination pair is randomly generated and a connection is set up between the newly generated source-destination pair. Since the source-destination pairs are randomly generated (each node is equally likely to be selected to form a pair), the traffic is uniformly distributed among different nodes. In addition, the constant bit rate (CBR) traffic model in ns-2 is employed for all the connections. Each simulation is executed for 1000 simulated seconds. We set the threshold δ in Eq. (2) to 0.1, for all the simulations.

Next, we evaluate the performance of our scheme.

5.2 Performance Evaluation

We organize this section as below. In Sections 5.2.1 through 5.2.3, we investigate how our incentive scheme performs under various number of connections, different data-rate of CBR traffic, and various dropping probability of selfish nodes, respectively. Section 5.2.4 shows the overhead incurred by our incentive scheme.

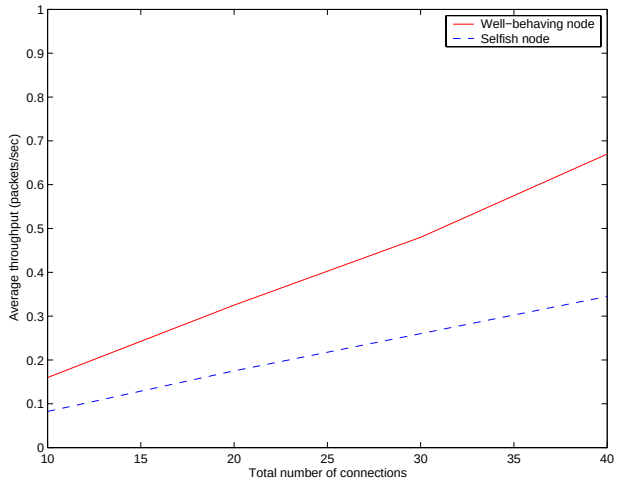

Figure 1: Throughput under various number of connections N_{conn}

5.2.1 Performance under Various Number of Connections

This experiment is to show the performance of our incentive scheme under various number of connections N_{conn}. In this experiment, a selfish node drops all the packets from other nodes unless it is the source or the destination of the packet; the data rate of all the CBR connections is fixed to 1 packet/sec.

Fig. 1 plots the average throughput of a well-behaving/selfish node vs. the total number of connections N_{conn}. Note that for each simulation run, the value of N_{conn} is fixed. In the figure, the average throughput of a well-behaving node is obtained by 1) summing up the number of packets correctly received by all well-behaving nodes, 2) dividing this sum by the total number of well-behaving nodes, and 3) dividing the result by the total simulation time, *i.e.*, 1000 seconds. Similarly, the average throughput of a selfish node is obtained by 1) summing up the number of packets correctly received by all selfish nodes, 2) dividing this sum by the total number of selfish nodes, and 3) dividing the result by the total simulation time.

As depicted in Fig. 1, a well-behaving node achieves significantly higher average throughput than that of a selfish node; on average, a selfish node suffers about 50% throughput reduction, as compared to a well-behaving node. This shows that our scheme can identify a

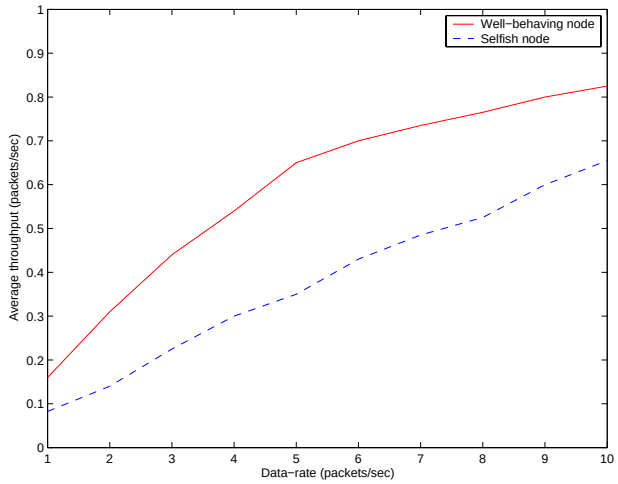

Figure 2: Throughput under various data-rate

selfish node and punish it accordingly.

5.2.2 Performance under Various Data-rate

This experiment is to show the performance of our incentive scheme under various data-rate of CBR traffic. In this experiment, a selfish node drops all the packets from other nodes unless it is the source or the destination of the packet. For each simulation, the data rate of all the CBR connections is fixed; but for different simulation, the data rate of CBR connections changes from 1 to 10 packets/sec. For all the simulations, we set the total number of connections $N_{conn} = 10$.

Fig. 2 shows the average throughput of a well-behaving/selfish node vs. the data-rate of a CBR connection. As shown in Fig. 2, a well-behaving node achieves higher average throughput than that of a selfish node; however, the difference (in terms of percentage) between the throughput of a well-behaving node and that of a selfish node reduces with the increase of the data-rate. The reason for this reduction is the following: as the data-rate increases, the probability of transmission collision increases, which results in higher probability of mis-calculation of objective reputation (refer to Remark 1).

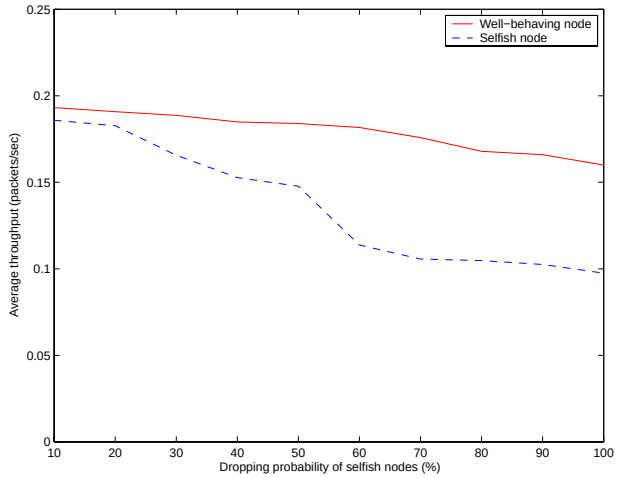

Figure 3: Throughput

5.2.3 Performance under Various Dropping Probability of Selfish Nodes

This experiment is to show the performance of our incentive scheme under various dropping probability of selfish nodes. In this experiment, each selfish node probabilistically drops the packets from other nodes unless it is the source or the destination of the packet. The dropping probability is the same for all selfish nodes and varies from 10% to 100% in different simulations. That is, for each simulation, the dropping probability of selfish nodes is fixed; but for different simulation, the dropping probability changes from 10% to 100%. For all the simulations, we set the total number of connections $N_{conn} = 10$ and fix the data rate of all the CBR connections at 1 packet/sec.

Fig. 3 plots the average throughput of a well-behaving/selfish node vs. the dropping probability of selfish nodes. It can be seen that as the dropping probability of selfish nodes increases, the gap between the throughput of a well-behaving node and that of a selfish node increases. Hence, our incentive scheme can not only distinguish the selfish nodes from the well-behaving nodes, but also impose a punishment proportional to the severity of the selfish behavior. This shows the effectiveness of our incentive scheme.

Fig. 3 also shows that the average throughput of a well-behaving node decreases with the increase of the dropping probability of selfish nodes. This is because the loss probability of

a packet destined to a well-behaving node increases with the dropping probability of selfish nodes.

From Fig. 3, one may notice that the average throughput of a selfish node is not zero when the dropping probability of selfish nodes is 100%. The reason is the following. When the dropping probability of a selfish node X is 100%, the overall evaluation record $OER_N(X)$ at node N is zero. Hence, $q = 1 - OER_N(X) = 1$ and $p = q - \delta = 0.9$ since we set δ in Eq. (2) to be 0.1. Therefore, the average throughput of a selfish node should $1 - p = 10\%$ of the transmission rate. From Fig. 3, we see the average throughput of a selfish node is about 0.1 packet/sec at a dropping probability of 100%, which is 10% of the transmission rate at 1 packet/sec. This validates our calculation.

One may argue that the average throughput of a selfish node should be zero when the dropping probability of the selfish node is 100%. This is the special case when $\delta = 0$, *i.e.*, zero tolerance about others' selfishness. As mentioned in Section 3.2, a dropping action could be occasionally triggered by some phenomena such as collision, rather than selfishness. If $\delta = 0$, two nodes may keep increasing the dropping probability and consequently fall into a retaliation situation. Letting $\delta > 0$ can help well-behaving nodes avoid this situation by treating each other a little bit more generously. Of course, the cost of non-zero δ is that a selfish node with a dropping probability of 100% can still have a throughput of $\delta \times \alpha$ (where α is its transmission rate), instead of zero; however, this throughput $\delta \times \alpha$ is small and a selfish node has to waste a good deal of energy to retransmit the dropped packets many times in order for the packets to get through to the destination. So, the node may even save energy if it cooperates in forwarding others' packets!

An open problem is how to tune δ so as to achieve the desirable trade-off between the throughput of a selfish node and the robustness of network stability (due to generosity in punishment). We leave this for future study.

5.2.4 Overhead Incurred by Our Incentive Scheme

This experiment is to show the overhead incurred by our incentive scheme, as compared to the scheme that does not use our incentive mechanisms. In this experiment, a selfish node

drops all the packets from other nodes unless it is the source or the destination of the packet; the data rate of all the CBR connections is fixed to 1 packet/sec. We do simulations for two schemes: one is our incentive scheme, and the other (which we call *benchmark scheme*) does not take any of the three actions, *i.e.*, neighbor monitoring, propagating reputation, and punishing selfish nodes.

Fig. 4 plots the average throughput vs. the total number of connections N_{conn}. The dashed line in the figure shows the average throughput of a node (averaged over all nodes) under the benchmark scheme (without incentive mechanisms); here, since the benchmark scheme does not distinguish the selfish nodes from the well-behaving nodes, the average throughput of a selfish node should be the same as that of a well-behaving nodes, from the statistical perspective. The solid line in the figure shows the average throughput of a well-behaving node under our incentive scheme. It can be observed that the throughput of a well-behaving node under our incentive scheme is reduced, as compared to that under the benchmark scheme. This throughput reduction is what we call *overhead*. The reason for this throughput reduction is two-fold: first, reputation propagation consumes bandwidth and therefore reduces the net throughput; second, collision may cause mis-calculation of the reputation measure, leading to improper punishment on the well-behaving nodes.

Fig. 4 shows that the overhead incurred by our scheme is not more than 8%, which is small. Just because of this small overhead, we are able to propagate reputation, identify selfish nodes and punish them according to the severity of their misbehavior. It can also be seen that the overhead increases with the total number of connections N_{conn}. This is because the larger N_{conn}, the high probability of collision, which results in higher probability of reputation mis-calculation and hence larger overhead.

6 Related Work

As mentioned in Section 1, the existing incentive schemes for packet forwarding can be classified into two categories, namely, *reputation-based* schemes and *pricing-based* schemes, the representative work of which are discussed as below.

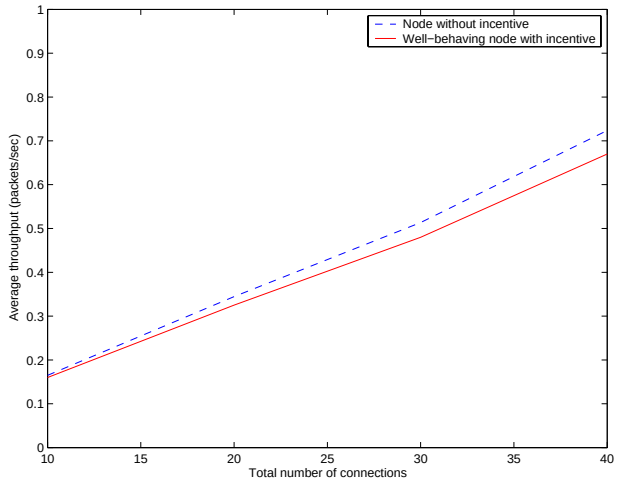

Figure 4: Communication overhead of our scheme

6.1 Reputation-based Schemes

One of the pioneering work on reputation-based incentive schemes is Ref. [8], where Marti et al. proposed two mechanisms called *watchdog* and *pathrater*, for non-cooperative ad-hoc networks. The *watchdog* identifies misbehaving nodes while the *pathrater* selects routes that avoid the identified misbehaving nodes. Their results showed that these two mechanisms can achieve an acceptable total throughput even with a large percentage of misbehaving nodes. However, how to punish a selfish node was not addressed in their work. Therefore, under their framework, selfish nodes are not discouraged and well-behaving nodes may be unfairly made busier. To remedy this, Buchegger and Le Boudec [1] proposed a punishment-based protocol called CONFIDANT. The nice feature of this protocol is that the nodes called *friends* can share the reputation information about other nodes, and punish the misbehaving nodes that have bad reputation. *Friends* play a crucial role in the protocol; however, how to win *friends* is not clear. In addition, encryption and digital signatures are used to protect the confidentiality and integrity of the communications among *friends*, but distributing (public) keys in ad-hoc networks is a challenging problem and is not addressed in their work.

In contrast to the above work, this paper proposed a secure reputation-based incentive architecture, in which not only a punishment mechanism is employed to discourage misbehavior, but also a set of secure protocols for identification, authentication, and routing are

specifically designed for 'non-cooperative' ad-hoc networks.

6.2 Pricing-based Schemes

Buttyan and Hubaux [2] proposed one of the early pricing-based incentive schemes. In their scheme, a virtual currency called *nuglet* is used as payments for packet forwarding. Later on, they [3] proposed another scheme based on credit counter. The nice feature of the two schemes is that they do not possess the limitation that many reputation-based schemes (excluding the SORI architecture in this paper) have: reputation may be falsified by selfish nodes. However, both of the schemes in [2, 3] require a tamper-resistant hardware module and assume that each node in the ad hoc network has almost the same amount of packets to forward. Moreover, the schemes introduce much management communication overhead to maintain the virtual currency flows.

Jakobsson et al. [5] proposed a nice scheme, which does not use tamper-resistant hardware. This scheme is designed for multihop cellular networks. Under the scheme, a micro-payment system suggested by Micali and Rivest [9] is employed, and the operator of the base stations plays the trusted role of banks in the micro-payment system. A disadvantage of this scheme is much communication overhead incurred due to managing the micro-payment system. For example, each time before a packet is forwarded at a node N, node N needs to send a request message to its neighbor and an acknowledgement from its neighbor is expected. To make things worse, if a packet reaches a node X and none of the neighbors of X is willing to forward the packet, the packet will be dropped and the previous forwarding efforts will be wasted.[7] In the scheme, whenever a base station receives a packet, the base station must contact the home network of the packet's originator to verify the message authentication code, which also introduces much communication overhead.

Pricing-based schemes require either tamper-resistant hardware [2] or virtual banks (trust authorities) that all parties can trust [5, 16]. As discussed in Section 1, these requirements impose serious limitations on the applicability of pricing-based schemes to non-cooperative

[7]This problem could be solved if the pricing query function in [5] is extended to some routing protocol such as the DSR.

ad hoc networks. In contrast, our incentive architecture is based on reputation and a belief that each node would like to stay in the network with a desirable quality of connectivity (punishment will seriously degrade the quality of connectivity of an identified selfish node). Our solution is comprehensive in the sense that it integrates reputation measurement, reputation propagation, punishment, node identification, message authentication, and secure routing. Each component in our incentive architecture is designed with careful consideration of the unique features of non-cooperative ad-hoc networks.

7 Concluding Remarks

In this paper, we propose a Secure and Objective Reputation-based Incentive (SORI) architecture to encourage packet forwarding and discipline selfish behavior in a non-cooperative ad hoc network. The unique features of our SORI architecture are the following.

1. The reputation of a node is quantified by objective measures (through neighbor monitoring).

2. The propagation of reputation is secured by a one-way-hash-chain based authentication scheme, which is computationally efficient and eliminates the need for a PKI or other forms of authentication infrastructures which are usually not practical for non-cooperative ad-hoc networks.

3. The reputation of a node is only propagated to its neighbors, which greatly reduces communication overhead as compared to the schemes that maintain reputation globally.

4. Routing is effectively secured with the aid of beaconing and the one-way-hash-chain based authentication scheme.

With the reputation measure obtained by the SORI architecture, we are able to design a punishment scheme to penalize selfish nodes. The experimental results show that the proposed scheme can successfully identify selfish nodes and punish them accordingly.

Acknowledgment

The first author would like to thank Drs. Adrian Perrig and Rohit Negi of Carnegie Mellon University for discussions related to this work.

References

[1] S. Buchegger and J. Le Boudec, "Performance analysis of the CONFIDANT protocol: cooperation of nodes - fairness in distributed ad-hoc networks," *IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Lausanne, Switzerland, June 2002.

[2] L. Buttyan and J. Hubaux, "Enforcing service availability in mobile ad-hoc WANs," *IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Boston, MA, USA, Aug. 2000.

[3] L. Buttyan and J. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks," *ACM/Kluwer Mobile Networks and Applications (MONET)*, vol. 8, no. 5, pp. 579–592, Oct. 2003.

[4] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communication & Mobile Computing,* Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications, vol. 2, no. 5, pp. 483–502, 2002.

[5] M. Jakobsson, J. Hubaux, and L. Buttyan, "A micro-payment scheme encouraging collaboration in multi-hop cellular networks," *Proceedings of Financial Crypto 2003*, Gosier, Guadeloupe, Jan. 2003.

[6] D. Johnson, D. Maltz, and J. Broch, "The dynamic source routing protocols for mobile ad hoc networks," *Internet Draft, IETF Mobile Ad Hoc Network Working Group*, Oct. 1999.

[7] L. Lamport, "Password authentication with secure communication," *Communications of the ACM,* vol. 24, no. 11, pp. 770–772, Nov. 1981.

[8] S. Marti, T. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," *Proceedings of Mobicom 2000*, Boston, MA, USA, Aug. 2000.

[9] S. Micali and R. Rivest, "Micropayments revisited," *Proceedings of the Cryptographer's Track at the RSA Conference 2002*, Bart Preneel (ed.), Springer Verlag CT-RSA 2002, LNCS 2271, pp. 149–163.

[10] P. Michiardi and R. Movla, "Simulation-based analysis of security exposures in mobile ad hoc networks," *European Wireless 2002 Conference*, Florence, Italy, Feb. 2002.

[11] NS developing group, "The network simulator – ns-2," available at http://www.isi.edu/nsnam/ns.

[12] P. Papadimitratos and Z. J. Haas, "Secure Routing for Mobile Ad hoc Networks", *Proceeddings of the SCS Communication Networks and Distributed Systems Modeling and Simulation Conference (CNDS 2002)*, 2002.

[13] A. Perrig, R. Canetti, D. Song, and J. Tygar, "Efficient and secure source authentication for multicast," *Proceedings of the Symposium on Network and Distributed System Security (NDSS 2001)*, Internet Society, 2001.

[14] T. S. Rappaport, *Wireless Communications: Principles & Practice,* Prentice Hall, 1996.

[15] C. K. Toh, "Associativity-based routing for ad-hoc mobile networks," *Wireless Personal Communications*, vol. 4, no. 2, pp. 1–36, Mar. 1997.

[16] S. Zhong, J. Chen, and Y. Yang, "Sprite: a simple, cheat-proof, credit-based system for mobile ad-hoc networks," *IEEE INFOCOM 2003*, San Francisco, CA, USA, April 2003.