

ASR: Anonymous and Secure Reporting of Traffic Forwarding Activity in Mobile Ad Hoc Networks

Heesook Choi, William Enck, Jaesheung Shin, Patrick McDaniel, Thomas F. La Porta
Department of Computer Science and Engineering
Pennsylvania State University, University Park, PA 16802
E-Mail: {hchoi,enck,jsshin,mcdaniel,tlp}@cse.psu.edu

Abstract

Nodes forward data on behalf of each other in mobile ad hoc networks. In a civilian application, nodes are assumed to be selfish and rational, i.e., they pursue their own self-interest. Hence, the ability to accurately measure traffic forwarding is critical to ensure proper network operation. These measurements are also often used to credit nodes based on their level of participation, or to detect loss. Past solutions employ neighbor monitoring and reporting on node forwarding traffic. These methods are not applicable in civilian networks in which neighbor nodes lack the desire or ability to perform the monitoring function. Such environments occur frequently in which neighbor hosts are resource constrained, or in networks where directional antennas are used and reliable eavesdropping is difficult or impossible.

In this paper, we propose a protocol that uses nodes on the data path to securely produce packet forwarding reports. Reporting nodes are chosen randomly and secretly so that malicious nodes cannot modify their behavior based upon the monitoring point. The integrity and authenticity of reports are preserved through the use of secure link layer acknowledgments and monitoring reports. The robustness of the reporting mechanism is strengthened by forwarding the report to multiple destinations (source and destination). We explore the security, cost, and accuracy of our protocol.

Index Terms

Ad Hoc Network, Security, Civilian Application, HMAC Chain.

I. INTRODUCTION

The establishment of a wireless infrastructure is non-trivial, especially in volatile environments where node mobility dominates. Occasionally, erecting fixed infrastructures is not feasible due to location or temporal validity. For example, it is not possible to build a wireless tower in the middle of a hostile battlefield. Furthermore, the tower cannot be moved as an attack progresses. In other mission-oriented scenarios such as search and rescue, terrestrial obstacles, e.g. avalanche prone mountains, inhibit the creation of fixed access points.

In the absence of a fixed infrastructure, mobile ad hoc networks (MANETs) can be used. By not requiring a fixed infrastructure or centralized control for communication, MANETs are well suited for the aforementioned scenarios. Within the network, multi-hop paths are created between nodes that formerly

could not communicate. Ideally, each node selflessly forwards each packet to the next node in the path. As nodes move, they leave and join various communication links, thus promoting many ephemeral paths.

Reliable operation in a MANET requires explicit cooperation between nodes. While this is feasible to assume for mission-oriented scenarios, careful consideration needs to take place when applying MANETs to civilian applications. In a civilian mobile ad hoc network, communicating nodes will use any relay points present. It is conceivable that selfish or malicious nodes exist in these networks. Additionally, reliability can be severely impacted by network congestion and mobility. Therefore, there is a need to detect selfish or malevolent behavior and promote cooperation between nodes.

One method for detecting malicious behavior is to generate reports on traffic flow between nodes. This information can be used to not only detect misbehavior, but also to indicate good network citizens. By identifying nodes that play fairly, a payment scheme can be implemented in order to further promote cooperation. Traffic reports can also be used to detect bottlenecks.

Previously proposed monitoring and reporting solutions rely on neighboring nodes to eavesdrop on data transmissions in order to generate reports. While this may work well in networks with trusted nodes, i.g., in military settings, it is not feasible for civilian ad hoc networks. Furthermore, such techniques may also fail in military settings if directional antennas are used, since nodes cannot reliably monitor data transmissions.

In this paper, we propose a secure random reporting protocol for a civilian ad hoc network, in which the source and destination collect reports from intermediate nodes on the routing path. Every delivered data packet initiates a report from one intermediate node that is randomly chosen by a source node. The chosen node then integrates its self-report into the packet before forwarding the transmission. We use a symmetric-key construction for selecting the reporting node that efficiently prevents disclosure of the selected node's identity from all adversaries except those that can mount large scale traffic analysis attacks. Note that reports may become lost due to mobility and congestion. In order to provide robustness in the face of loss, the report is sent to the either source, destination, or both.

Because only the selected node modifies the report field, eavesdropping nodes may observe incoming and outgoing packets of a node and determine when a node has generated a report. To thwart this attack, an efficient report wrapping scheme is proposed along with the secure random reporting protocol.

While the secure random reporting protocol provides secret node selection, as well as integrity and authenticity of reports, it does not guarantee that the self-report is accurate. Although nodes cannot manipulate others' reports, they may not be trusted to generate accurate reports. To rectify this inadequacy, we propose a forgery detection scheme that provides proofs of delivery implemented by secure network layer acknowledgments.

We have simulated our approach using ns-2 [6]. Our results show that we accurately monitor packet forwarding activity even in lossy networks. We further simulate malicious packet dropping to look at the effectiveness of our secure random reporting protocol.

The rest of this paper is organized as follows. Section II reviews previous research in malicious node detection and cooperation in ad hoc networks. Section III describes possible threats in civilian ad hoc networks. Section IV presents an overview of the proposed random reporting protocol. This scheme is then strengthened in Section V as we extend it to provide report integrity, node selection confidentiality, and prevention of falsified reports. Next, Section VI provides simulation results and computational overhead of the secure random reporting protocol. Finally, Section VII concludes.

II. RELATED WORK

Detection of malicious behavior and collection of cooperation history for crediting are two motivating factors for monitoring nodes. This section discusses previous research in these areas.

A. *Detection of Malicious Behavior*

The Watchdog/Pathrater [12] scheme proposes the use of a watchdog for detecting misbehaving nodes, and a pathrater to help the routing protocol avoid detected misbehaving nodes. The design utilizes intermediate nodes along the routing path, wherein a node sends a packet to an intermediate downstream

node and verifies the node that forwards it. If the node does not send the packet within a predefined period, it is declared as misbehaving, and the monitoring node notifies the source. Pioneering the area of intrusion detection in ad hoc networks, Zhang and Lee [20], [21] propose a general architecture, in which all nodes participate in the monitoring of data transmission. Each node is responsible for monitoring a transmission range and cooperating with neighboring nodes in order to detect intrusions. Zhang and Lee later proposed a second scheme to reduce the number of nodes involved in monitoring [1]. In this cluster-based scheme, a cluster head (CH) is elected for monitoring data traffic within the transmission range. The elected CH is responsible for monitoring all neighboring nodes and checking statistics. AODVSTAT [18] implements an intrusion detection system (IDS) within the AODV [15] routing protocol. The system monitors for routing message drops, data-packet drops, MAC/IP spoofing, and resource depletion attacks. In AODVSTAT, an IDS monitors all observable transmissions from neighbors. Note that all of the above schemes require some level of communication eavesdropping. These solutions are not feasible in our target environments because reliable eavesdropping is not possible.

Awerbuch et al. [2] propose an alternate scheme that uses intermediate nodes on the data path. If a source does not receive an ACK from a destination, the source begins probing all intermediate nodes. This causes each node along the path to send an ACK back to the source. Unfortunately, due to the dynamic characteristics of MANETs, data paths can change frequently, possibly before the failed link is found.

B. Cooperation

Many times, cooperation between nodes cannot always be expected without incentives. Several algorithms have been proposed that use payment schemes. A node may be paid via a credit for behaving cooperatively or excluded/penalized for misbehaving.

Sprite [23] proposes an incentive system where selfish nodes are encouraged to cooperate. In Sprite, each node is motivated to honestly report its actions, even in the presence of selfish node collusion. Intermediate nodes retain receipts of received messages. The receipt is then sent to the CCS (Credit Clearance Service) as proof of forwarding, and the CCS then charges/credits based on the received reports.

CORE [14] uses a collaborative reputation mechanism to encourage nodes to cooperate. The reputation is calculated via both direct and indirect observation by a node and its neighboring nodes, respectively, within the transmission range. In similar scheme, CONFIDANT [3], each node monitors nodes existing one hop away. If a node detects and concludes malice, it generates an ALARM message to either a source or a friend. This, in turn, causes misbehaving nodes to be excluded from the community.

All of the aforementioned detection and cooperation schemes require the observation of neighboring nodes. Additionally, these schemes deal only with detection or cooperation. Our reporting protocol targets more general applications, including both detection of malicious behavior and crediting for cooperation. The information provided by the reporting scheme is also vital for detecting data bottlenecks.

III. THREAT MODEL

When considering civilian ad hoc networks, there is a potential for malicious behavior from self-interest and maliciousness. This section discusses these threats and how they pertain to packet forwarding activity and report collection. The discussion illuminates the set of threats to which we aim to be resilient.

It is important to note that the high loss and delay prevalent in wireless and mobile networks exacerbates the problem of detecting selfish/malicious nodes. If a node drops packets and moves, it is difficult to detect whether the packet loss is from mobility or selfishness/maliciousness. Likewise, in a congested network, packets are dropped because of packet buffer overflows. Distinguishing between selfish/malicious drops and congestion is difficult. Regardless of the reasons for packet loss, a source node may wish to avoid particular nodes due to the mere occurrence of lost packets, whether it be the result of selfish/malicious behavior or simply network congestion.

Most forms of non-cooperation result in *denial of service* (DoS). In the extreme case, an ill-performing node would simply refrain in participating in routing, and hence would never be placed on a path. Possibly more damaging, a similar attack would allow the node to accept a position on the path, but it would not forward data packets. Our protocol does nothing to prevent or detect attacks on the routing protocol, but rather focuses on accurate reporting of packet forwarding.

Nodes may also drop packets selectively. For example, a selfish node may choose not to forward packets for a specific source or destination, or conversely, simply favor a source or destination by dropping traffic for others when they are in competition. Similarly, the node can choose particular applications to drop or show preferential treatment. Finally, a node may randomly drop packets in order to simply save energy.

Note that only a few well-selected drops are necessary to vastly reduce the throughput between a source and destination: each drop causes the congestion control algorithm to aggressively throttle traffic [7]. Connection recovery is slow, and the attacker gains advantage with little effort [19].

More subtle attacks exist. In credit based systems, a node benefits from forwarding more packets than its neighbors. To gain an advantage, a malicious node injects fake packets. This expends the energy of all forwarding nodes, thereby rendering them incapable of forwarding future legitimate packets. The known defense for this attack is to use interleaved hop-by-hop authentication schemes [24], [25], where fake packets are filtered mid-transmission. This paper does not address this type of attack.

Existing proposed cooperation schemes for civilian ad hoc networks use rewards or penalties to encourage cooperation. Rewards and penalties are dictated by reports of mobile node behavior. The credit for relaying other traffic might be money, more bandwidth, or higher priority service. The policy motivates mobile nodes to cheat, manipulate, or drop the reports so that they get more credit and avoid being penalized. Defending against potential forwarding and replay attacks on the reporting data is a challenging issue for monitoring the packet forwarding activity.

IV. OVERVIEW OF RANDOM REPORTING PROTOCOL

For the purposes of this paper, it is assumed that dynamic source routing (DSR) [9] is used. The DSR routing protocol provides a full path between the source and destination. It is reasonable to assume that the source and destination nodes are trusted, as they are the entities responsible for the data traffic. The protocol focuses on the secure reporting of forwarding activities for the data transmission.

Each intermediate node only needs to keep track of its own contribution, instead of observing the actions of other nodes. Using intermediate nodes in this manner is rational when dealing with a civilian

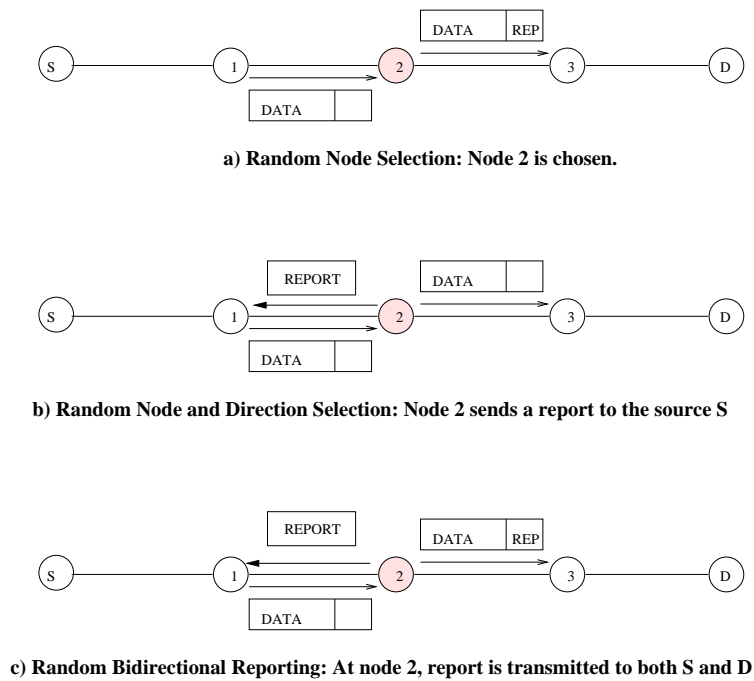


Fig. 1. Random Reporting Protocol: source S and destination D

ad hoc network. The rest of this section provides an overview of the Random Reporting Protocol. While alone the Random Reporting Protocol is not secure, Section V introduces the Secure Random Reporting Protocol.

One naive approach for generating self reports is that each node on a path periodically sends a report to the source/destination of a flow. This simple periodic reporting scheme functions well for static networks, but it does not work well for dynamic networks, or networks with malicious nodes. First, the scheme's quality is highly dependent on report transmission frequency. Additionally, rapid changes due to mobility or congestion quickly degrade its effectiveness, because reports may be lost or paths may change before reports are gathered. Since reliable transmission is not guaranteed, the disappearance of a node's report may cause it to be incorrectly viewed as an anomalous or congested point, even if it has correctly forwarded all data packets. The report data is transmitted via the same path as normal data. This allows a selfish or malicious node to know the source of any report. The node can then drop or change particular reports for their own self interest.

A. Random Reporting Node Selection (RRNS)

In order to address the aforementioned problems with packet manipulation and dynamic networks, we propose a Random Reporting Node Selection (RRNS) method. For every packet, the source randomly chooses one intermediate node to send a report to the destination. This is accomplished by coupling each data packet with a report, so that when the report is received by the destination, the relaying activity of intermediate nodes can be dynamically observed.

In RRNS, if the path consists of n intermediate nodes, any node can be chosen with probability $1/n$. Figure 1-(a) illustrates RRNS where node 2 has been randomly chosen. In general:

- 1) For all packets p , source S randomly chooses (uniform distribution) intermediate node n_i to send a report. S attaches a report request RR to p , identifying n_i as the chosen node.
- 2) For a packet p with RR for n_i , n_i attaches report R to p before forwarding to destination D .
- 3) Destination D receives p , including R from all intermediate nodes, and periodically analyzes the reports, looking for traffic deviations.

The idea of choosing a random node is motivated by micro-payment [8], [13], in which a randomly chosen transaction is used for a merchant to deposit some amount of money. Applied to RRNS, the randomly chosen intermediate node should add to the forwarding packet the number of packets it has forwarded since joining the path.

Since the intermediate node is selected randomly, other nodes are unable to predict the selection schedule. This prevents nodes from timing their attacks to maximize their duration. While the randomness provides better reports, the described scheme is vulnerable to attack. Without taking precautions, reports may be manipulated by downstream nodes with selfish intentions. Section V addresses this by introducing secret node selection. In summary, RRNS is advantageous, because it gathers reports from nodes in real time and has very low communication overhead, due to the coupling of reports with every data packet. We quantify this overhead in Section VI.

B. *Random Reporting Node and Direction Selection (RRNDS)*

In RRNS, if packets are lost due to congestion or mobility, the destination will only receive the reports sent before the anomaly occurred. If malicious intermediate nodes are located close to the destination and drop packets which select nodes before them, the destination may receive reports only from these malicious nodes. Thus, the destination may misinterpret the location of the problem.

Random Reporting Node and Direction Selection (RRNDS) is proposed to make RRNS more robust. RRNDS extends Step 2 of RRNS by randomly deciding the direction to send the report at the chosen node. If the report is sent towards the destination, it is attached to the data packets, just as in RRNS. On the other hand, if a source-bound direction is chosen, a separate report message is transmitted, or it may be piggybacked on traffic on the reverse path. Figure 1-(b) shows this scheme.

C. *Random Bidirectional Reporting (RBR)*

The report in RRNDS is transmitted to either the source or the destination. Unfortunately, the amount of report information received by the destination or source is reduced in RRNDS. Due to this shortage of reports, the source or destination may not precisely analyze the relaying activity of intermediate nodes.

We address this problem by modifying Step 2 of RRNS to transmit the report to both the source and destination. This technique, shown in Figure 1-(c), is referred to as Random Bidirectional Reporting (RBR). In the figure, node 2 sends a report to the destination and source node. Simulation results reported in Section VI show that bidirectional reporting improves effectiveness in the face of mobility.

Additionally, for both RRNDS and RBR, if the communication between source and destination is bidirectional, source-bound reports are attached to data packets destined for the source. This reduces communication overhead.

V. SECURE REPORTING PROTOCOL

The random reporting protocols discussed in Section IV are based upon random node selection. If intermediate nodes (selfish or malicious) discover a packet including a report and the selected node, the

information may be manipulated or dropped.

This section proposes an efficient construction that conceals the node selection from other intermediate nodes. In civilian mobile ad hoc networks, the intermediate nodes cannot be assumed to be honest; lying may provide more credit. Thus, in order to assure the validity of node reports, a chain of HMACs on the link layer acknowledgments is proposed. This addition provides forgery detection.

The following notations are used in the secure reporting protocol and forged report detection schemes.

- ID_i : Identifier of node n_i .
- K_{ij} : a pair-wise key between node n_i and n_j .
- $hash(x)$: Cryptographic hash function computation for x
- σ : $HMAC(K_{SD}, DATA|ID_i)$ computation result for the data and ID_i .
- $DATA$: Data transmitted between the source and destination.
- R_f : Report for the forward traffic.
- R_b : Report for the backward traffic.
- H_R : HMAC result over forward and backward reports, $H_R = HMAC(K, R_f|R_b)$.
- $E_{K_{ij}}(X)/D_{K_{ij}}(X)$: Encryption/Decryption for X with a symmetric key K_{ij}

Mobile devices are less powerful in computation and have a battery of limited lifetime. Therefore, symmetric cryptography is often more appropriate for mobile devices in ad hoc networks. The secure random reporting protocol requires three pairs of symmetric keys: source and destination, source and intermediate nodes, and destination and intermediate nodes. Schemes for distributing symmetric keys in ad hoc networks have been proposed [11], [5], and thus will not be discussed. Any efficient symmetric key management scheme can be used for distributing symmetric keys to mobile nodes; its choice does not impact our results.

A. Secure and Random Reporting Protocol

DSR allows the source node to know the full routing path. The source node chooses one intermediate node n_i uniformly at random, and computes *Token*, which is added to the data packet. The *Token*

contains the node selection information which is not disclosed. The use of an *HMAC* in the computation of the *Token* provides randomness and secrecy in the node selection. The selected node identifier may be contained in the encrypted data so that a destination easily discovers it by decrypting the data packet, while other nodes cannot. The source performs the following operations:

Source:

- Choose one intermediate node n_i
 - Compute $\sigma = \text{HMAC}(K_{SD}, \text{DATA}|ID_i)$,
 - Compute $H_i = \text{hash}(K_{Si}|\sigma)$
 - Generate $\text{Token} = \sigma \oplus H_i$
- first intermediate node: $[\text{DATA}, \sigma, \text{Token}]$

When a node receives a packet, it needs to determine if it is the randomly selected node. At the same time, no other nodes can be allowed to know which node has been chosen. Upon receiving a data packet $(\text{DATA}, \sigma, \text{Token})$, an intermediate node n_j computes $H_j = \text{hash}(K_{jS}|\sigma)$ and XORs it with the received *Token*. If the result of the XOR operation is equal to the received σ , the node knows it was chosen. This is only satisfied at node n_i since the source used a pairwise key K_{Si} . Since the above test in other intermediate nodes is not satisfied, they do not generate reports.

The chosen intermediate node n_i sends its report by attaching it to the data packet. The report R includes the number of packets the node forwarded for the source and destination. For integrity purposes, the chosen intermediate node computes *hash* with its report R and its pair-wise symmetric key shared with the destination.

$$H_D = \text{hash}(K_{iD}|R)$$

$$\text{Report} = [R, H_D]$$

This scheme is resilient to another node n_k replacing $\text{Token} = \sigma \oplus H_i$ with $\sigma \oplus H_k$, because the resulting $\text{HMAC}(K_{SD}, \text{DATA}|ID_k)$ is not equal to the received $\sigma = \text{HMAC}(K_{SD}, \text{DATA}|ID_i)$. Without knowing K_{SD} , the node n_k cannot change σ to masquerade as a selected node. When receiving a

TABLE I
RANDOM AND SECURE REPORTING

<p>Source:</p> <ul style="list-style-type: none"> - Choose one intermediate node n_i - Compute $\sigma = HMAC(K_{SD}, DATA ID_i)$ - Compute $H_i = hash(K_{Si} \sigma)$ - Compute $Token = \sigma \oplus H_i$ - Send the packet $(DATA, \sigma, Token)$
<p>Intermediate Node n_i:</p> <ul style="list-style-type: none"> - Compute $H_i = hash(K_{iS} \sigma)$ - XOR H_i with $Token \rightarrow H_i \oplus Token = H_i \oplus \sigma \oplus H'_i$, where H'_i is received - Check if $XOR(Token, H_i) == \sigma$ - If n_i is chosen, <ul style="list-style-type: none"> o Compute $H_D = hash(K_{iD} R)$ o Generate $Report = [R, H_D]$ and attach it to the data packet \rightarrow next hop node: $[DATA, \sigma, Token, Report]$
<p>Destination:</p> <ul style="list-style-type: none"> - Evaluate if $hash(K_{Di} R) = H_D$ to find out the chosen node. - Save the report and check whether there exists a misbehavior. - If the report is not valid, ignore the report.

data packet, the destination checks that σ and the received report R are valid, i.e. if $hash(K_{Di}|R) = H_D$ and $\sigma = HMAC(K_{DS}, DATA|ID_i)$ are satisfied.

From a security point of view, if the selected node generates an extra-packet, neighboring nodes may learn that the node generates a report. To deal with this problem, a data packet has two report fields: forward report and backward report. For example, the node selected by a destination-bound packet generates reports for both the forward and backward flows. The forward report field contains a report that the selected node relayed for the source-to-destination flow and the backward report field includes a report for the opposite flow. Now, the selected node computes the HMAC for both report fields once. *Report* has the format $[R_f, R_b, H_R = HMAC(K_{iD}, R_f|R_b)]$. The source-bound report is processed in a similar way. The source-bound packet has $[R_f, R_b, H_R = HMAC(K_{iS}, R_f|R_b)]$, where R_f and R_b are the reports for the destination-to-source traffic flow and source-to-destination traffic flow, respectively.

The key idea of node selection is to conceal the node selection from other nodes. Table I summarizes the basic idea of secure random reporting protocol.

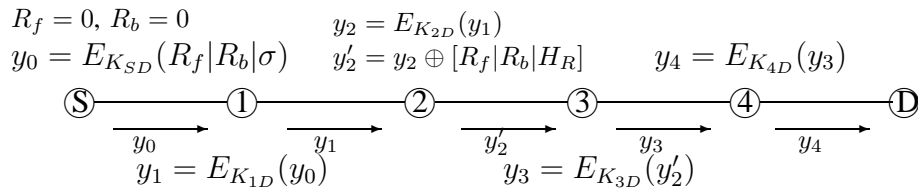
B. Report Wrapping Scheme

While the secure random reporting protocol protects the identity of the selected node from in-path adversaries, it is susceptible to traffic analysis. If an adversary can overhear traffic going in and out of node n_i , determining whether or not n_i was selected is trivial. If the incoming and outgoing data does not match, n_i has attached a report. Therefore, if this adversary is downstream from n_i , it can selfishly strip out the report.

In the Internet, a great deal of research has been done to provide anonymous communication using a proxy function (Mix, Jundo, and Onion Router) [4], [17], [16]. Different approaches [10], [22] in ad hoc networks have been proposed, considering features such as mobility, congestion, and energy and computation limitations. These existing solutions aim to provide anonymous communication for data packets. Our goal is to provide anonymity of the reporting node from eavesdropping nodes. We propose an efficient transformation scheme in which eavesdropping nodes may not discover whether a node generates a report.

Every node on the routing path encrypts the received report field, $y_i = E_{K_{iD}}(y_{i-1})$ where y_{i-1} is a received report field generated by the previous node. Unlike other intermediate nodes, the selected node first encrypts the report field as other nodes, and XORs the encrypted report field with its report value. Figure 2 shows how each intermediate node processes the report field. The encryption key that nodes use is a symmetric key shared with the destination node. Although the initial value of R_f and R_b at a source node is 0, σ which is a message authentication code makes the encrypted result random. The encrypted report field is a random value to other nodes en route and eavesdropping nodes, since they don't know the symmetric key of other nodes.

A destination knows all the nodes en route and the selected node (n_i). The destination can generate the



(a) Report transformation in each intermediate node

$$\begin{array}{l}
 y_2 = E_{K_{2D}}(E_{K_{1D}}(E_{K_{SD}}(0|0|\sigma))) \\
 y'_2 = D_{K_{3D}}(D_{K_{4D}}(y_4)) \\
 [R_f|R_b|H_R] = y_2 \oplus y'_2 = y_2 \oplus y_2 \oplus [R_f|R_b|H_R]
 \end{array}$$

(b) Operations at destination node D

Fig. 2. Report transformation against traffic analysis attack

encrypted report field value ($y_i = E_{K_{iD}}(y_{i-1})$) by repeatedly encrypting the report field, from a source to a selected node in sequence, with symmetric keys. At the same time, it decrypts the received report field until it recovers the report field transmitted by the selected node, $R_i' = y_i \oplus [R_f|R_b|H_R]$. The destination node XORs two values ($y_i \oplus R_i'$) which outputs the report $[R_f|R_b|H_R]$ generated by the selected node n_i .

The scheme above explains how the destination-bound report field is processed in every node. For the source-bound report field, the operations every node processes are the same as the destination-bound report. Let z_i denote the source-bound report field that a node n_i generates. The report field is $z_i = E_{K_{iS}}(z_{i-1})$.

C. Report Forgery Detection Scheme

Even if the report transmission is secure, we still need a way to validate the report provided by the selected node. To address this, a report forgery detection scheme is proposed. This scheme combats both a report forgery attack and some colluding attacks.

In wireless networks, the link layer provides an acknowledgment (ACK) by which a receiving node confirms to a sending node that it received the packet successfully. The sending node waits for an ACK from the receiver. If it does not receive an ACK after retransmitting a packet several times, it considers the link to the receiving node broken and sends a ROUTE ERROR message back to the source node. If this occurs, the source node will change the data path.

This property of the link layer protocol is used to provide forgery detection of reports. Each data packet

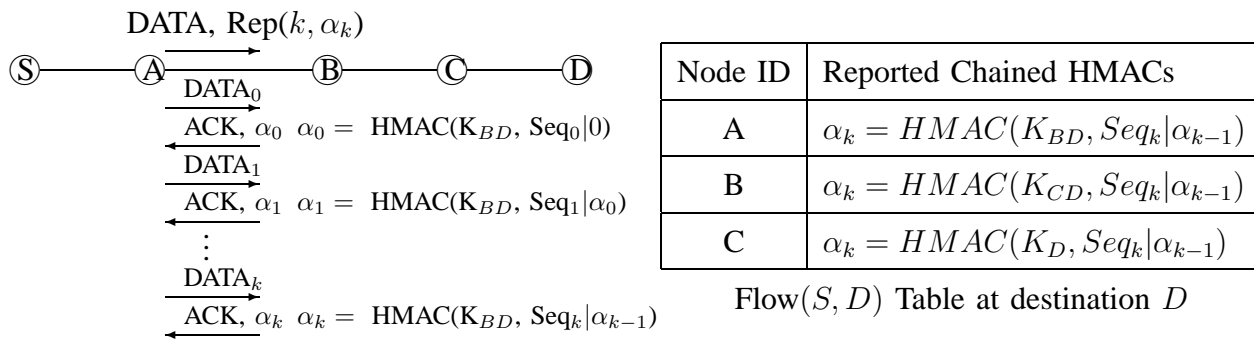


Fig. 3. Chained HMACs to Detect Report Forgery

is sent in a link layer frame. For each data packet i , successfully received in frame j_f , the receiver sends an ACK(j_f, seq_i) and α_i , which is an HMAC of α_{i-1} and seq_i . Intermediate nodes generate self-reports for each flow defined by the source and destination addresses. Intermediate nodes may relay data packets for multiple flows. They generate ACKs with HMAC chains for the packet sequence number pertaining to the flow.

For the HMAC's symmetric key, the receiver R uses the pair-wise K_{RD} shared with the destination. The forgery detection scheme does not use a key shared between two neighboring nodes in order to prevent these nodes from colluding and allowing one node to manipulate the HMAC.

Figure 3 provides an example communication flow of the report forgery detection scheme. Data transfer begins with initial packet DATA0. For this **first packet**, node B computes $\alpha_0 = \text{HMAC}(K_{BD}, seq_0|0)$. Node B then sends to A both a link layer ACK and the computed α_0 . After the initial packet, node B uses the previous HMAC, α_{i-1} , in the computation of α_i .

$$\alpha_i = \text{HMAC}(K_{BD}, seq_i|\alpha_{i-1})$$

When node A is randomly chosen to send a report for data packet k , both k and α_k are transmitted. In the case of Figure 3, since the report direction is towards the destination, the report is attached to a DATA transmission.

Mobility is fundamental to MANETs. When a path changes, the destination may not have the initial sequence number for a particular path. Therefore, for the **first report** in a path, node A must include α_0 ,

the initial sequence number, α_i , the sequence number map, and the number of packets it has forwarded. Since the destination knows the symmetric key, K_{BD} , it can verify α_0 was created correctly. The most recent HMAC, α_i , can then be verified by computing the HMAC chain from α_0 .

Single malicious node: The chained HMACs are intended to detect if nodes forge their reports. The forgery detection scheme protects against two types of misbehavior. In Figure 3, node A may try to cheat by sending a report saying it forwarded 100 packets when it really only forwarded 50 packets. In order for A to cheat under the described scheme, it must provide α_{100} . Since generation of α_{100} requires knowledge of K_{BD} , only known by node B and D , A cannot fake the report. The only way for A to learn α_{100} is for node B to tell it. This only occurs after A forwards B 100 packets.

The forgery detection scheme also protects against a malicious node B . In one case, node B may purposefully choose to not send an ACK to node A . If this occurs, node A will send a ROUTE ERROR to the source node, and a new data path will be chosen. If, on the other hand, node B sends A an ACK, but purposefully generates the wrong HMAC, for the initial α_0 or the intermediate α_i , node A will report the wrong α_0 or α_i to the destination.

Fortunately, the destination can determine if α_0 or α_i is incorrect. Since the destination can compute $HMAC(K_{BD}, seq_0|0) = \alpha_0$, if the resulting α_0 does not match the received α'_0 , it can detect that node B is misbehaving. Similarly, the destination can compute α_k and compare it to the received α'_k . As shown in Figure 3, the destination retains knowledge of the state by keeping a table consisting of the node identifier and the most recently received HMAC, α_i . The destination determines the expected α_k by calculating the HMAC chain.

$$\alpha_k = HMAC(K_{BD}, seq_{k-1} | \dots HMAC(K_{BD}, seq_{i+1} | \alpha_i))$$

If the received α'_k does not match the computed α_k , the destination can determine if B is behaving maliciously.

To support both source/destination-bound reports, a receiving node generates and attaches α_k and β_k to an ACK, where β_k is $HMAC(K_{BS}, seq_k | \beta_{k-1})$ in the example above which is computed with the

symmetric key to the source.

Another concern is that malicious nodes may replay reports. The proposed secure random reporting protocol collects reports from intermediate nodes en route, keeping track of the routing path of packets. The addition of a sequence number counters replay attacks. To ensure packets in different sessions have different sequence numbers, the source uses a combination of the session identifier and packet sequence number. This prevents report replay, as the report validation scheme uses the sequence number field to compute α_k .

Colluding nodes: In addition to preventing report forgery, the chained HMAC scheme prevents nodes in collusion from dropping packets. Nearby colluding nodes may exchange information to determine the random node selection. There are two collusion scenarios: by non-adjacent nodes and by adjacent nodes. In the former scenario, if a node that is not in collusion and located between two colluding nodes is selected to generate the report, the colluding nodes may drop the packet and generate a false report that they transmit. In this case, malicious behavior is easily detected since the colluders do not have the chained HMACs generated by the cooperative node.

In the second colluding scenario, adjacent nodes, e.g. node 2 and 3 in Figure 2 (a), may collude to drop packets and generate reports as if they forwarded all the packets. Since they may release their keys to each other, this colluding attack may result in forged report packets, i.e., the chained HMACs may appear to contain valid information. The last colluding node, however, which is adjacent to a cooperative node, cannot generate a report stating that it forwarded all the packets to the non-colluding node. In order to generate this report, the node needs α_k that can be generated only by the cooperative node when it receives k packets. Hence, the last colluding node will be detected by the forgery detection scheme. The source and destination nodes will not use paths that include this node. While detection of compromised nodes that are not at the end of the colluding group may not be possible initially, if these nodes remain on the path, they will eventually be detected using the same techniques. To limit a colluding attack by adjacent nodes, other approaches such as game theoretic schemes [23] may be used to encourage nodes

not to collude. This is out of the scope of this paper. Section VI examines the resulting computational overhead.

VI. RELIABILITY AND COMPUTATIONAL OVERHEAD

In order to analyze the reliability of the proposed reporting schemes, we simulated our protocols using ns-2. These simulations illustrate the robustness of the RBR and other related schemes. Additionally, we explore the cost of the underlying cryptographic constructions based on empirical data.

A. Simulation Environment

The experimental testbed is as follows. Table II shows the parameters of the ns-2 simulations. Mobile nodes use IEEE 802.11 MAC with a transmission range of 250m. Additionally, the CMU scenario generation tool [6] was used to create a network consisting of 50 mobile nodes in an 1500X300 range. A random waypoint mobility model with speeds of 20 was used. m/sec.

We examine the effects of mobility and traffic loads on three secure random reporting protocols. We run simulations with 4 different pause times to study the effect of mobility. We include 25 and 30 CBR background traffic flows in our simulations. All points are the averaged value over 5 runs.

TABLE II

SIMULATION PARAMETERS

Simulation Time	900 seconds
Number of nodes	50
Packet Size	512 bytes
Mobility	Random waypoint mobility model and 20 meter/sec.
Routing Protocol	Dynamic Source Routing (DSR)
Transport Protocol	UDP

The observation period is a fixed time during which the destination and source observe the reports to analyze the network properly. A basic observation period of ten seconds was used. Traffic flows are defined by the source and destination addresses. In order to adapt to the dynamic characteristic of path changes, the reports were collected based on the flow and path. This is required because a flow may

change its path due to a link failure caused by mobility or congestion. As the path changes, the source and destination keep track of the flow state, the current path state, and the active path list that consists of paths transmitting the flow traffic during the observation period.

When a node encounters a link failure and sends a RERR (ROUTE ERROR) message to the source, the appropriate sequence number and report are included in the notification. Until the source receives the RERR, it continues to use the current path to transmit data. Therefore, all packets containing sequence numbers higher than the notified packet on the path causing the RERR are lost.

B. Simulation Results and Discussions

Packet loss occurs due to mobility, congestion, and malicious dropping. Identifying the source of packet loss is difficult due to the random nature of its occurrence. The effectiveness of the protocol (shown in the following figures) is the percentage of experiments that correctly identify the malicious nodes. In this conservative model, the adversary behaves only slightly different than well behaved nodes: nodes that more aggressively drop packets will be detected more easily, and the protocols will be more effective.

The simulation was performed with two different attack strategies for dropping packets: fixed dropping rate (simple attack: Case 1) and by matching the malicious dropping rate with the naturally occurring average dropping rate for the network under its current conditions (Case 2). For each case, we devised a simple detection algorithm which is described below. The detection algorithms themselves are not important; they are for illustrative purposes only to quantify the impact of the report protocol variants.

Case 1: We set the dropping rate of a malicious node to a fixed value. We measured the average packet loss rate over the simulation time, excluding malicious nodes. We had ten background traffic sources generate 4 packets/sec and a target source generate about 28 packets/sec. Results showed a 12% average packet loss (caused by congestion and mobility) from this baseline test. Using the measured average packet loss as a guideline, a second battery of experiments simulated an adversary that dropped 17% of the received packets at different speed; we chose a slightly higher value (17%) than the average (12%) to

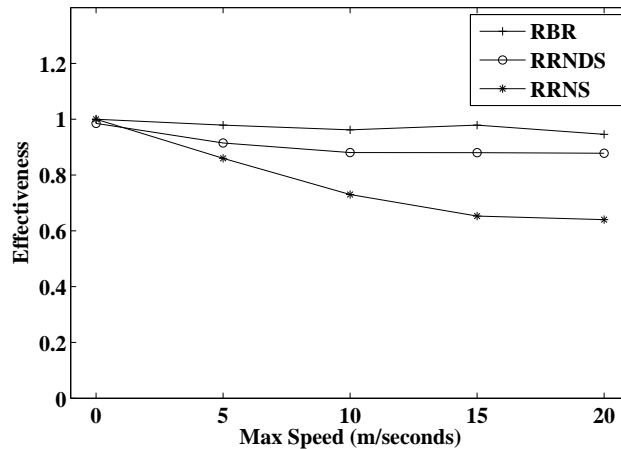


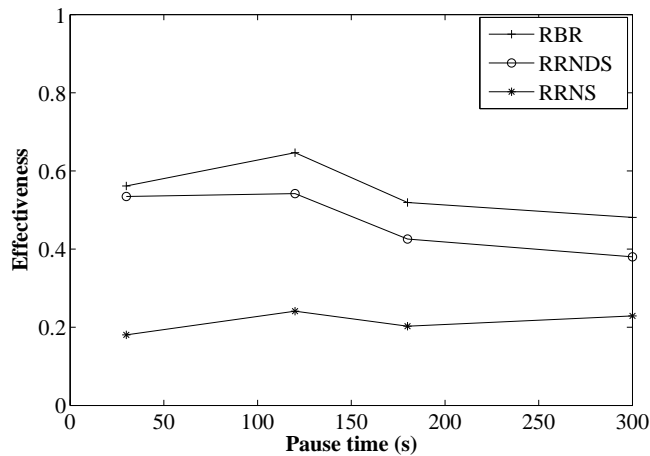
Fig. 4. Effectiveness vs. Mobility in a fixed attacking rate

accommodate the variance of packet loss rate. We designated any node that the reports showed to have a loss rate greater than 12% as anomalous.

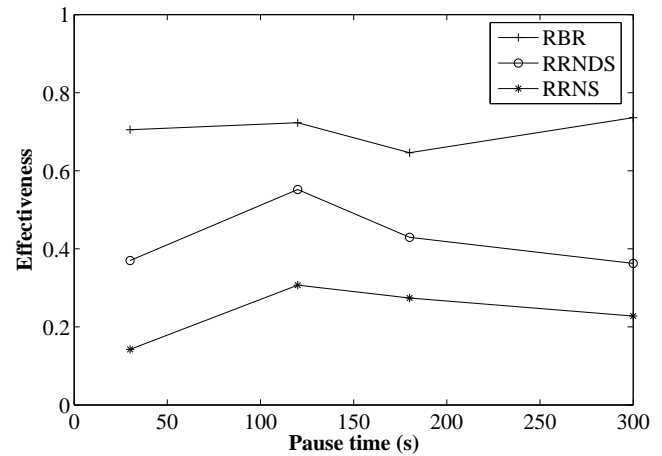
Figure 4 shows the impact of mobility on the effectiveness of the three protocols under this simple attack. In the static case, all three protocols showed an almost 100% detection rate. However, as mobility is introduced, the effectiveness of RRNS decreases sharply since the reports embedded in data packets are lost. By contrast, the RBR protocol remains highly effective in all experiments. In RBR, the report is transmitted to both the source and destination. The redundant transmission improves the robustness of reports in the presence of mobility. We will discuss the effectiveness of RRNDS below.

Case 2: With low traffic load and low mobility, if a malicious node drops too many packets, a source or destination may easily detect malicious behavior. Since paths change frequently due to high mobility or congestion, it is not easy to estimate how long a path will be used for data transmission. Considering this, we assume that an adversary may have information about the network statistics and use it to schedule packet drops. One policy that an adversary may employ is to drop packets at the rate of the average packet drops due to natural congestion or mobility. This way, an adversary may reduce the possibility of being detected.

To simulate this adversary model, we evaluated average packet loss rate and its standard deviation at



(a) 25 CBR Sources



(b) 30 CBR Sources

Fig. 5. Effectiveness vs. Mobility/Loads

each scenario, leaving out malicious drops. The average packet loss rate has a different value at four pause times (30, 120, 180, and 300 seconds) under 25 and 30 CBR background traffic sources. Every traffic source generates 4 packets/sec. A designated malicious node en route dropped packets at the average packet loss rate measured at a specific condition. The detection algorithm in the source and destination used the average packet loss rate plus its standard deviation to detect the malicious dropping. Although more sophisticated algorithms can be used to detect the malicious dropping, we picked this simple detection algorithm to evaluate the robustness of three random reporting protocols for illustration purpose.

Figure 5 shows the effectiveness of three random reporting protocols under the intelligent attack strategy, and with 25 and 30 CBR background traffic sources respectively. The effectiveness is degraded as mobility decreases. In a stable network (low mobility and low traffic load), the average packet loss rate is very low, and thus adversary's dropping is also very low. Under this condition, the subtle packet dropping is not easily distinguishable from the normal packet loss. By the same reasoning, the effectiveness under 30 background traffic sources is better than under 25 background sources.

In both attack strategies, RRNDS showed higher effectiveness than RRNS even though the source and destination receive only about half of the reports, i.e., the same total number of reports are received. This can be explained by the additional information that a source node has. In addition to source-bound

reports, the source node knows how many packets it transmitted and receives some useful information from RERRs, by which the source can evaluate the number of lost packets by a link failure on the path. The destination only receives reports included in the successfully received packets.

To evaluate the effect of the extra information, we simulated a scenario with a 120 second pause time. In this case, the source received almost half of the reports and 137 RERRs in RRNDS. RRNDS was 1.4 times as effective as RRNS. To confirm the effect of the additional information, we also simulated a special case in which all reports are transmitted only to the source. In this case, the source receives successful reports and extra information from RERRs. This improves the effectiveness of RRNDS and RRNS by 56% and 280% respectively, but is still lower than RBR's effectiveness. While sending all reports to the source appears to be advantageous, this is vulnerable to a report dropping attack by downstream nodes. When the communication is uni-directional (source-to-destination), the selected node has to generate an extra packet to send a report to the source node. A downstream node of the selected node knows that the packet is a report and drops the packet. Moreover, in the case of bi-directional communication, as the effectiveness of RBR shows, the reports transmitted to both direction can help the detection algorithms in the source and destination.

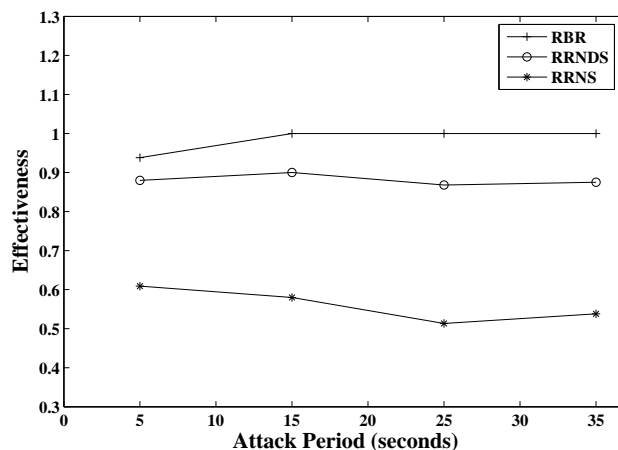


Fig. 6. Effectiveness in Random Reporting Protocols

We further simulated adversaries mounting attacks of variable length. As shown in Figure 6, RRNS

was the least effective. Most of the undetected drops in RRNS occurred where the source changes to a new path, one intermediate node on the new path drops packets, and then the data path changes due to the following link failure. This does not allow any reports to reach the destination, and the forwarding activity cannot be discovered. The results also show that RRNDS improves RRNS's effectiveness by 150%. The RBR protocol achieves near perfect effectiveness under attacks of all lengths.

C. Overhead

The secure random reporting protocol requires three new fields: *Token*, *Report*, and HMAC chain. Both the *Token* and HMAC chain are the same size as the cryptographic hash output (16 bytes for MD5). The *Report* is composed of the number of forwarded packets for forward and backward flows (2x4 bytes) and its cryptographic hash output (16 bytes). The total overhead incurred by the secure random reporting protocol is 56 bytes which is only 3% of maximum data packet size. On the other hand, the existing eavesdropping schemes require a separate packet transmission to inform other nodes of the reports. This separate packet transmission incurs additional transmission delay and energy consumption. Our reporting protocol removes all these extra costs by embedding a report in data packet with small overhead in packet size.

One possible concern of the Secure Random Reporting Protocol is the computational overhead. Depending on the chosen cryptographic function, the overhead will vary. To test the performance of HMAC and encryption/decryption, we wrote a module for the Linux 2.6 kernel, using the available cryptographic API. Tests were performed on a Pentium 3 800MHz, 192MB RAM system using a stock Linux 2.6.11 kernel compiled by GCC 3.3. The tests covered MD5, SHA1, and SHA256 digest functions with varying key sizes (64bit, 128bit, 160bit, and 256bit). AES CBC encryption/decryption is also evaluated with varying key size (128bit, 192bit, and 256bit). All obtained results were the average raw cycle count over 1000 test runs. The high number of tests runs was chosen to ensure more accurate results.

In our simulation, the longest path has ten intermediate nodes. For this longest path, we estimate the computational overhead of the secure reporting protocol. As expected, there was no performance difference

for varied key sizes, therefore, the average raw cycle count for the key sizes was used. Finally, using the `cpu_khz` value of 647894, actual time latencies were calculated.

Table III shows the computation time for individual calculations, which emulated the actual data used by the protocol. This shows the results of the case that a report field includes forward and backward report values and the report is transmitted to both the source and destination. As described earlier, the input data for each stage of the HMAC chain consists of a sequence number and the output of a previous HMAC. Thus, the total input data size for the HMAC chain is four bytes for the sequence number and the number of bytes outputted by each cryptographic digest function (MD5 = 16 bytes, SHA1 = 20 bytes, SHA256 = 32 bytes). The HMAC data column in the table represents the overhead for performing the HMAC on the Maximum Transmission Unit (MTU), 1500 bytes.

In the report wrapping scheme, every intermediate node encrypts the report field, and a destination performs encryption and decryption of that field. The report field (40 bytes) consists of 2 reports (forward and backward: $2 * 4$ bytes) and a hash computation result of two reports (32 bytes in SHA256). AES encryption/decryption of the report field takes $2.7905 \mu s$. This turns out to be $55.81 \mu s$ overhead.

TABLE III

HMAC COMPUTATIONAL OVERHEAD (647,894K CYCLES/SEC)

Algorithm	HMAC Chain	HMAC Data	Total
MD5	$7.037 \mu s$	$29.950 \mu s$	$271.01 \mu s$
SHA1	$19.108 \mu s$	$86.614 \mu s$	$746.468 \mu s$
SHA256	$20.308 \mu s$	$95.606 \mu s$	$800.452 \mu s$

The total computational cost consists of four factors: two HMACs of data packets at the source and destination, two chained HMACs (α_k and β_k) at each intermediate node, a hash computation (*Token* checking) at each of the intermediate nodes and the destination, and encryption/decryption of the report field at each node and the destination. According to the results of computational overhead above, the secure random reporting protocol, report wrapping and forgery detection schemes have less than $856.262 \mu s$ overhead. Under optimal circumstances, it takes 65 milliseconds to transmit data from the source to

the destination (10 intermediate nodes). The total HMAC, hash, and AES computation takes only 1.3% of the total time ($856.262\mu s/65.856ms = 0.856262/65.856$).

VII. CONCLUSIONS

Most military applications of MANETs target mission oriented scenarios such as battle fields and emergency rescue. In these scenarios, mobile nodes actively cooperate with each other to achieve a goal. This is different than civilian mobile ad hoc networks, where nodes are not necessarily cooperative.

In this paper, we propose an anonymous and secure random reporting protocol for a civilian ad hoc network, in which the source and destination collect reports from intermediate nodes on the routing path. Every data packet initiates a report from one intermediate node that is randomly chosen by a source node. Through a symmetric cryptographic construction, we ensure that the node selection is not disclosed to other intermediate nodes.

We devise a chained HMAC scheme on the link layer acknowledgments to verify the validity of the received report. Furthermore, an efficient report wrapping scheme is proposed to prevent eavesdropping nodes from learning the reporting node selection by analyzing the report field going in and out of a node.

From both security and performance perspectives, the secure random reporting protocol is advantageous for gathering the forwarding activities of mobile nodes in civilian ad hoc networks. The protocol has a small communication overhead due to the increase in packet size caused by including the real time reports with data transmission. Our simulation results demonstrate the promising possibility of the reporting protocol.

VIII. ACKNOWLEDGEMENTS

The work is supported by National Science Foundation (NSF) grant number CNS-0519460.

REFERENCES

- [1] Y. an Huang and W. Lee. A Cooperative Intrusion Detection System for Ad Hoc Networks. *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks(SASN)*, 2003.

- [2] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An On-Demand Secure Routing Protocol Resilient to Byzantine Failures . *ACM WiSe*, 2002.
- [3] S. Buchegger and J.-Y. L. Boudec. Performance Analysis of the CONFIDANT Protocol(Cooperation Of Nodes: Fairness in Dynamic Ad-hoc NeTworks). *MOBIHOC*, 2002.
- [4] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonym s. *Communications of the ACM*, 1981.
- [5] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. *ACM Conference on Computer and Communication Security*, 2002.
- [6] <http://www.isi.edu>. The Network Simulator - ns-2, 2000.
- [7] P. J. Transmission Control Protocol - DARPA Internet Protocol Program Specification. *IETF*, Sep. 1981. RFC 793.
- [8] M. Jakobsson, J.-P. Hubaux, and L. Buttyan. A Micro-Payment Scheme Encouraging Collaboration in Multi-Hop Cellular Networks. *In Proceedings of Financial Cryptography*, 2003.
- [9] D. B. Johnson, D. A. Maltz, Y.-C. Hu, and J. G. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (DSR). <http://www.ietf.org/internet-drafts/draft-ietf-manet-driETF> draft, 2004.
- [10] J. Kong and X. Hong. ANODR:ANonymous On Demand Routing with Untraceable Routes for Mobile Ad-hoc Networks. *In ACM MOBIHOC*, 2003.
- [11] D. Liu and P. Ning. Establishing Pairwise Keys in Distributed Sensor Networks. *ACM Conference on Computer and Communication Security*, 2003.
- [12] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. *Proc. of ACM Mobicom*, 2000.
- [13] S. Micali and R. Rivest. Micropayments Revisited. *CT-RSA*, 2002.
- [14] P. Michiardi and R. Molva. CORE: A Collaborative Reputation Mechanism to enforce node cooperation in Mobile Ad Hoc Networks. *In Proceedings of The 6th IFIP*, 2002.
- [15] C. E. Perkins and E. Belding-Royer. Ad hoc On-Demand Distance Vector (AODV) Routing. *IETF RFC3561*, 2003.
- [16] M. G. Reed, P. F. Syverson, and D. M. Goldschlag. Anonymous Connections and Onion Routing. *Journal on Selected Areas in Communication Special Issue on Copyrig ht and Privacy Protection*, 1998.
- [17] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
- [18] G. Vigna, S. Gwalani, K. Srinivasan, E. Belding-Royer, and R. Kemmerer. An Intrusion Detection Tool for AODV-based Ad hoc Wireless Networks. *20th Annual Computer Security Applications Conference*, 2004.
- [19] X. Zhang, S. Wu, Z. Fu, and T.-L. Wu. Malicious Packet Dropping: How It Might Impact the TCP Performance and How We Can Detect It. *In Proceedings of ICNP 2000*, Nov. 2000.

- [20] Y. Zhang and W. Lee. Intrusion Detection in Wireless Ad Hoc Networks. *6th International Conference Mobile Computing and Networks*, 2000.
- [21] Y. Zhang and W. Lee. Intrusion Detection in Wireless Ad Hoc Networks. *Proc. of ACM Mobicom*, 2000.
- [22] Y. Zhang, W. Liu, and W. Lou. Anonymous Communications in Mobile Ad Hoc Networks. *IEEE INFOCOM*, 2005.
- [23] S. Zhong, J. Chen, and Y. R. Yang. Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks. *Proc. of IEEE INFOCOM*, 2003.
- [24] S. Zhu, S. Setia, S. Jajodia, and P. Ning. An Interleaved Hop-by-Hop Authentication Scheme for Filtering of Injected False Data in Sensor Networks. *In Proc. of IEEE Symposium on Security and Privacy*, 2004.
- [25] S. Zhu, S. Xu, S. Setia, and S. Jajodia. LHAP: A Lightweight Hop-by-Hop Authentication Protocol For Ad-Hoc Networks. *In Proc. of the 23rd International Conference on Distributed Computing Systems Workshops*, 2003.