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 traffic forwarding of nodes. 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.

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

I. Introduction

The establishment of a wireless infrastructure is non-trivial, especially in volatile environments in which node mobility dominates. Occasionally, erecting fixed infrastructures is not feasible due to location or temporal validity. 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 both mission oriented and civilian applications. 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 available. It is conceivable that selfish or malicious nodes exist in these networks. Therefore, there is a need to detect selfish or malevolent behavior and promote cooperation between nodes.

To deal with the selfishness of nodes in civilian ad hoc networks, researchers have proposed many solutions. These solutions encourage nodes to cooperate either by remunerating cooperative behavior or by penalizing malicious and selfish behavior [26], [17], [4], [3], [15]. Most of these solutions are based on information gathered through monitoring of neighbor's behavior.

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 or are malicious, nodes can better choose with whom to cooperate.

In order for these techniques to function properly, it is imperative to securely and reliably collect traffic reports. Previously proposed monitoring and reporting solutions rely on neighboring nodes to eavesdrop on data transmissions of other nodes in order to generate reports. While this may work well in networks with trusted nodes, e.g., in military settings, it is not feasible for civilian ad hoc networks. Considering the selfish nature of nodes in a civilian network, we cannot expect that they are willing to monitor others and collect the information to report. Furthermore, such reporting techniques assisted by neighboring nodes may not work well if directional antennas are used. Unlike the omnidirectional antennas, the coverage area of directional antennas is of a sector shape and this limits the ability of eavesdrop.

To address these problems, we propose an anonymous and secure random reporting protocol (ASR) in which contributions of intermediate nodes on data traffic forwarding are securely and reliably collected. In ASR, intermediate nodes on a path generate a self-report of traffic forwarding: 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 data packet. We use a symmetric-key construction for selecting the reporting node that efficiently prevents disclosure of the selected node's identity. 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.

While the basic 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.

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

We qualitatively analyze the security that ASR affords against single node misbehavior and colluding nodes. We have simulated our approach using ns-2 [8]. Our results show that ASR accurately monitors packet forwarding activity even in lossy networks. We further simulate malicious packet dropping to determine the effectiveness of ASR. Finally, we analyze the performance and overhead of ASR.

The rest of this paper is organized as follows. Section II describes possible threats in civilian ad hoc networks and assumptions that we use throughout this paper. Section III reviews previous research in malicious node detection and cooperation in 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. We qualitatively analyze security of ASR in Section VI. Section VII provides simulation results and overhead of the secure random reporting protocol. Finally, Section IX concludes.

II. Model and Assumptions

A. Threat Model

Civilian ad hoc networks are prone to self interest and malicious behavior. The most straightforward threat to civilian networks is a *denial of service* (DoS). For example, a misbehaving node may simply refrain from participating in routing. However, in such a case, the node is never placed on a path. A possibly more damaging attack occurs when a node acquires a position on a path, but selectively drops packets to degrade performance. We address this latter attack. Our protocol does not aim to prevent or detect attacks on the routing protocol (e.g., the former case), but rather focuses on secure reporting of packet forwarding (e.g., the latter case). Since the forwarding activities are dictated by reports, nodes may attempt to misrepresent themselves by creating, manipulating, or dropping reports in order to veil their misbehavior. For example, a node that maliciously drops packets will specifically wish to drop packets reporting its behavior.

Malicious or selfish nodes can also collude in creating or dropping reports. This collusion exists in different forms: collusion between nodes not on a routing path, collusion between nodes on a path and nodes not involved in forwarding, collusion between non-adjacent nodes on a path, and collusion between adjacent nodes on a path. In the first type, colluding nodes eavesdrop communications and claim that they cooperatively have forwarded packets. In the second form, nodes on a path may give collected traffic information to other nodes not on a path and generate a report. Similarly, non-adjacent nodes on a path can collude in exchanging information to hide their misbehavior. Finally, adjacent selfish or malicious nodes on a path can easily collude in hiding the existence of their packet drops.

More subtle attacks also exist. To gain an advantage, a malicious node may inject 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 [27], [29], in which fake packets are filtered mid-transmission. This paper does not address this attack.

B. Assumptions

We assume that there is no centralized server or Internet connection in MANETs. In this setting, the source and destination must protect their own flows from being disrupted by malicious nodes. Therefore, we assume that the source and destination police their own flows. They collect reports and determine which node is behaving maliciously for their flow. Based on the collected information, they can change to a new routing path which does not include misbehaving nodes, refrain from being using paths containing these nodes in the future, or refuse to forward packets for these nodes. The source and destination's actions, once they discover a malicious node, are out of the scope of this paper.

For the purposes of this paper, we assume the use of a source routing protocol such as dynamic source routing (DSR) [12] so that the full path information is available. The full path information is used as a basis to verify which nodes participate in forwarding packets. In this paper, we propose a secure random reporting scheme in which intermediate nodes on a path generate reports of their cooperation on forwarding data traffic. We assume that routing protocol security is provided by some other means [9]. We also assume that good-behaving nodes follow the rules of the control and routing protocols.

Throughout the paper, our protocol is based on symmetric key cryptosystems. We assume that there exists an efficient key management scheme to establish pairwise keys. Symmetric cryptography is appropriate for ad hoc networks, due to the limited computational power and battery power of mobile devices. Key management has been actively studied in ad hoc and sensor networks, such as probabilistic key predistribution [7], [5], [28] and symmetric polynomial based key establishment [14]. The key management is a separate important issue and out of scope of this paper.

III. Related Work

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

A. Detection of Malicious Behavior

The Watchdog/Pathrater [15] 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 uses intermediate nodes along the routing path, wherein a node sends a packet to an intermediate downstream node and verifies that this node forwards the packet. If the node does not forward the packet within a predefined period, it is declared as misbehaving, and the monitoring node notifies the source.

Zhang and Lee [23], [24] 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 neighbors in order to detect intrusions. Zhang and Lee later proposed a second scheme to reduce the number of nodes involved in monitoring [10]. 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 [22] implements an intrusion detection system (IDS) within the AODV [18] 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. A node may be paid via a credit for behaving cooperatively or excluded/penalized for misbehaving. We classify the existing solutions into three classes: credit-based, penalty-based, and utility function based schemes.

In the first class, relay nodes are remunerated for forwarding packets for others. Zhong et al. [26] proposed an incentive system, named Sprite, in which 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) connected to the Internet as proof of forwarding, and the CCS charges/credits based on the received reports. CORE [17] 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. Buttyan and Hubaux [4] proposed a stimulation scheme in which a node can transmit its packet only when it has enough credit count, called *nuglet*. Nodes in the network can earn nuglets by forwarding packets for others. The authors presented four rules that a node might take and showed that the node could receive the best performance when it follows a cooperative rule.

In the second category, selfish and misbehaving nodes are penalized for their non-cooperative behavior by being excluded from the routing and community. In CONFIDANT [3], each node monitors one-hop neighbor nodes. 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. The Watchdog/Pathrater [15] scheme also belongs to this group.

In the third class, a utility function is defined at each node such that nodes can maximize the utility function (their benefit) by cooperatively forwarding packets for others. These solutions are based on the mathematical framework of game theory. Srinivasan et al. [21] proposed a distributed acceptance algorithm. In this algorithm, relay nodes maintain the history of services that they receive and provide. The authors proposed a game theoretic strategy (Generous TIT-FOR-TAT) which is used for each node to decide whether it accepts a new relay request based on the past history. Anderegg and Eidenbenz [1] proposed a routing protocol, called Ad hoc-VCG, based on game theory. In the Ad hoc-VCG, during route discovery, each node makes its energy cost and transmission power known to other nodes, and a destination chooses the most cost-efficient path based on the collected information. In the Ad hoc-VCG mechanism, nodes cannot receive higher payments by cheating, which encourages nodes to relay packets of others.

In our system, we assume that there is no centralized controller or trusted third party as there is in Sprite. The reports of traffic forwarding activities are collected by the endpoints of a flow, i.e., the source and destination. These nodes can use the reports to detect misbehaving nodes on their own active flows, and thus take immediate actions to protect their flows. In addition, the reports may be used with some of the reputation based systems discussed above.

IV. Overview of the Random Reporting Protocol

In this section, we present a basic random reporting protocol. The key idea is that each intermediate node only needs to keep track of its own contribution, instead of observing the actions of other nodes. This use of intermediate nodes is appropriate for a civilian ad hoc network. By introducing randomness, it is more difficult for an adversary to discover, delete, or modify reports. Furthermore, to address packet manipulation and network dynamics, we propose three random reporting protocols: Random Reporting Node Selection (RRNS), Random Reporting Node and Direction Selection (RRNDS), and Random Bidirectional Reporting (RBR). We present these protocols here to describe the basic reporting operations and motivations. We use this protocol as the basis of the secure random reporting protocol of Section V. We detail these protocols in the following subsections.

A. Random Reporting Node Selection (RRNS)

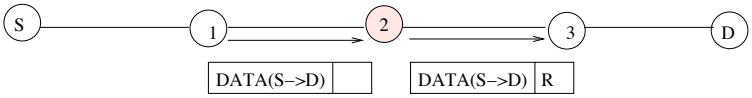
For every data packet, the source randomly chooses one intermediate node to generate a report to the destination. This is accomplished by coupling each data packet with a report, therefore when the destination receives the packet, the relaying activity of intermediate nodes can be dynamically observed.

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:

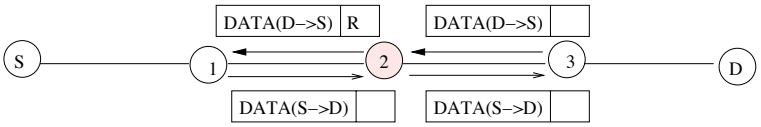
1) For every data packet p, source S randomly chooses (uniform distribution) an intermediate node n_i to generate a report for this flow. S attaches the identifier of randomly chosen node to the packet.
2) For a packet p with selected node n_i, n_i attaches report R to p and forwards p carrying report R towards destination D.
3) Destination D receives p, maintains and periodically analyzes the forwarding activity of all intermediate nodes involved in forwarding, looking for traffic deviations.

The idea of choosing a random node is similarly used by [11], [16] for micro-payment, in which a randomly chosen transaction is used for a merchant to deposit some amount of money.

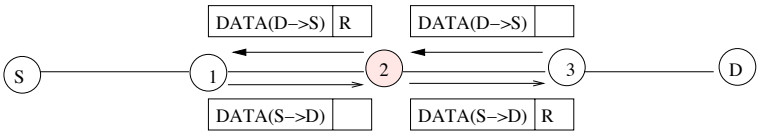
Since the intermediate node is selected randomly, other nodes are unable to predict the selection schedule. While the randomness provides better reports, the described scheme is vulnerable to attack. Without taking precautions, reports may be manipulated by downstream nodes with selfish intentions. Section V addresses this by introducing a secret node selection scheme.



a) Random Node Selection: Node 2 is chosen.



b) Random Node and Direction Selection: Node 2 sends a report to the source S



c) Random Bidirectional Reporting: At node 2, report is transmitted to both S and D

Fig. 1. Random Reporting Protocol: source S and destination D: node 2 is selected to generate a report

B. Random Reporting Node and Direction Selection (RRNDS)

In RRNS, if malicious intermediate nodes are located close to the destination, they may drop packets which contains reports from upstream nodes. In this case, the destination may receive reports only from these malicious nodes and 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 allowing the chosen node to decide the direction to send the report. 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, the report may be piggybacked on traffic on the reverse path, or a separate report message is transmitted. 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, this reduces the amount of report information received by the source or destination. This shortage of reports may cause the source or destination to imprecisely 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 VII show that bidirectional reporting improves effectiveness in the face of mobility. Finally, just as in RRNDS, 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 efficient constructions that conceal the node selection from other intermediate nodes, and provide a forgery detection scheme.

The following notation is used in the following subsections to describe the anonymous secure reporting protocol.

- 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 of node n_i, $H_R = HMAC(K_{iD}, R_f|R_b)$.
- $E_{K_{ij}}(X)$/$D_{K_{ij}}(X)$: Encryption/Decryption for X with a symmetric key K_{ij}

A. Secure and Random Reporting Protocol

Based on the path information, 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. Using an $HMAC$ in the computation of the $Token$ provides both randomness and secrecy in the node selection. The selected node identifier is 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: choose one intermediate node n_i and encrypt data with the selected node identifier, $E_{K_{SD}}(DATA|ID_i)$; compute $\sigma = HMAC(K_{SD}, DATA|ID_i)$; compute $H_i = hash(K_{Si}|\sigma)$; generate $Token = \sigma \oplus H_i$; send a packet, $[DATA, \sigma, Token]$, to the first intermediate node.

When a node receives a packet, it needs to determine if it is the randomly selected node. Upon receiving a data packet, $[DATA, \sigma, Token]$, an intermediate node n_j computes $H_j = hash(K_{jS}|\sigma)$ 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 uses a pairwise key K_{Si} to compute the Token. 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 has forwarded for the flow. This scheme is resilient to report manipulation in which node n_k replaces $Token = \sigma \oplus H_i$ with $\sigma \oplus H_k$, because the resulting $HMAC(K_{SD}, DATA|ID_k)$ is not equal to the received $\sigma = HMAC(K_{SD}, DATA|ID_i)$. Without knowing K_{SD}, the node n_k cannot change σ to masquerade as a selected node. When receiving a data packet, the destination checks that σ and the received report R are valid, i.e. if $\sigma = HMAC(K_{DS}, DATA|ID_i)$ is satisfied.

If the selected node generates an extra-packet to send the report to the source, this will be visible to neighboring nodes. To deal with this problem, the report field consists of two subfields: forward report (R_f) and backward report (R_b) which indicate the number of forwarded packets for destination-bound and source-bound traffic flows, respectively. A destination-bound packet selects a node to generate both forward (source to destination) flow and backward (destination to source) flow reports. Similarly, a source-bound packet is processed such that *Report* contains $[R_f, R_b]$.

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

B. Report Forgery Detection Scheme

Even if the reporting node selection is secure, we still need to assure that a report is truthful. To address this, a report forgery detection scheme is proposed.

In many wireless networks, such as 802.11, link level acknowledgments (ACK) are used to help overcome the losses on the wireless links due to transmission errors or mobility. We used the link level acknowledgments to provide information for the forgery detection.

The ACK allows a receiving node to confirm to a sending node that a packet has been received successfully. If the sending node does not receive an ACK from the receiver after several retransmissions, the link is considered broken. In many common protocols, such as DSR, an error message, called Route Error, is sent back to the source node. If this occurs, the source node will change the data path.

Each data packet 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) consisting of seq_i, α_i, and β_i. α_i is an HMAC for path p, the packet sequence number (seq_i), and α_{i-1}, while β_i indicates an HMAC for path p, the packet sequence number (seq_i), and β_{i-1}.

The receiver r uses pairwise keys K_{rS} and K_{rD} as the symmetric key for α and β respectively, since the report is transmitted to both the source and 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. This chained HMAC requires intermediate nodes to only maintain current state information, not the history of packet transmissions.

Figure 2 provides an example flow of chained HMACs for the report forgery detection scheme. Suppose that data transfer begins with initial packet $DATA_0$. For this first packet, node B computes $\alpha_0 = HMAC(K_{BD}, p|seq_0|0)$ and $\beta_0 = HMAC(K_{BS}, p|seq_0|0)$. Node B then sends to A a link layer ACK carrying the computed α_0 and β_0. After the initial packet, node B uses the previous HMACs, α_{i-1} and β_{i-1}, in the computation of α_i and β_i. That is,

$$\alpha_i = HMAC(K_{BD}, p|seq_i|\alpha_{i-1})$$
$$\beta_i = HMAC(K_{BS}, p|seq_i|\beta_{i-1})$$

In the case of Figure 2, 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, nodes may not have the sequence number for a particular path. Moreover, a node on a path may selectively drop packets. Hence, it is necessary to include a sequence number map (*smap*) in the report so that the source and destination receiving the report can determine if it is correct. The sequence number map is a bit representation of forwarded packets, which starts with the sequence number (seq_s) of the first packet received after the previous report has been generated. For example, suppose that after generating a report, node n_i receives packets having sequence numbers 3 ($seq_s = 3$), 4, 5, 8, and 10. The sequence number map is "11100101" (left to right).

The report generated by an intermediate node is composed of the number of forwarded packets (*Pkts*=5), the sequence number (seq_s=3), the sequence number map (*smap*), and the latest α_i. In summary, the reports have the following format: $[Pkts, seq_s,\ \text{smap}, \alpha_i]$.

Since the destination knows the key, K_{BD}, it can verify α_i was created correctly. The most recent HMAC, α_i, can then be verified by computing the HMAC chain from the initial α with the initial sequence number and sequence map information.

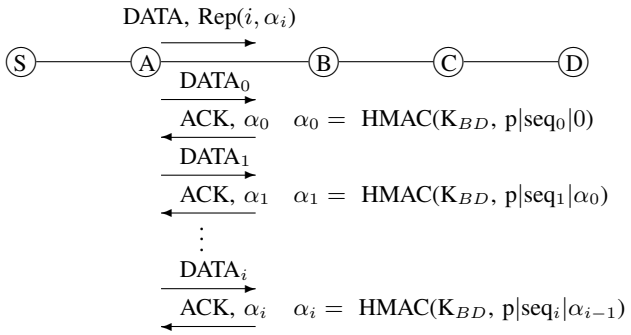
C. Report Wrapping Scheme

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.

TABLE I

RANDOM AND SECURE REPORTING

Source:
- Choose one intermediate node n_i and encrypt, $E_{K_{SD}}(DATA\|ID_i)$
- Compute $\sigma = HMAC(K_{SD}, DATA\|ID_i)$
- Compute $H_i = hash(K_{Si}\|\sigma)$
- Compute $Token = \sigma \oplus H_i$
- Send the packet $(DATA, \sigma, Token)$
Intermediate Node n_i:
- 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,
o Generate $Report = [R]$ and attach it to the data packet
$\rightarrow$ next hop node: $[DATA, \sigma, Token, Report]$
Destination:
- Check the integrity of data packet and decrypt it 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.



Node ID	Reported Chained HMACs
A	$\alpha_i = HMAC(K_{BD}, p\|seq_i\|\alpha_{i-1})$
B	$\alpha_{i+1} = HMAC(K_{CD}, p\|seq_{i+1}\|\alpha_i)$
C	$\alpha_{i+2} = HMAC(K_D, p\|seq_{i+2}\|\alpha_{i+1})$

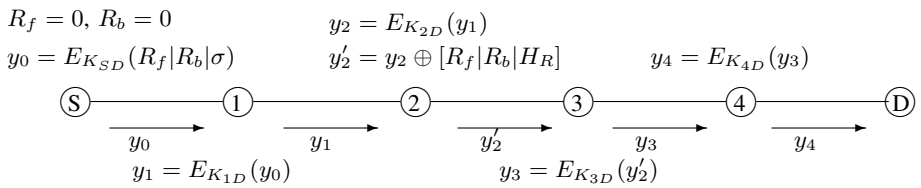
Flow(S, D) Table at destination D

Fig. 2. Chained HMACs to Detect Report Forgery: Suppose that A, B, and C are selected to generate a report in sequence for packets seq_{i+1}, seq_{i+2}, and seq_{i+3}, respectively.

A great deal of research has been done to provide anonymous communication in the Internet using a proxy (Mix, Jundo, and Onion Router) [6], [20], [19]. Different approaches [13], [25] have been proposed in ad hoc networks, 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, $y_i = E_{K_{iD}}(y_{i-1})$ where y_{i-1} is the received report generated by the previous node. In this way, the report field is altered by every node on a path. The encrypted report field appears random to other nodes. Unlike other intermediate nodes, the selected node first encrypts y_{i-1} and XORs the encrypted value with its report value. Figure 3 shows how each intermediate node processes the report field. Nodes use a pairwise key with the destination as the encryption key. 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.

A destination knows all the nodes en route and the selected node (n_i). The destination can generate the 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. 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 the two values ($y_i \oplus R_i$') which outputs the report $[R_f|R_b|H_R]$ generated by the selected node



(a) Report transformation in each intermediate node

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

(b) Operations at destination node D

Fig. 3. Report transformation against traffic analysis attack: node 2 is selected to generate a report

n_i.

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

VI. Security Analysis

In this section, we qualitatively analyze the security that ASR provides against the misbehavior of both single node and colluding nodes on reporting of traffic forwarding activities. Primarily, we analyze selfish behavior that harmfully affects the reporting scheme. We begin by looking at attacks from a single node, and then progress to situations of multiple node collusion.

A. Single Node Misbehavior

The goal of a single malicious node is to be placed on a forwarding path, and then to drop packets to degrade the performance of the flow. The malicious node will attempt to evade discovery as long as possible. To do this, it may drop, manipulate, or fabricate reports to hide its behavior. In this subsection, we present how ASR combats against these single node attacks.

In ASR, intermediate nodes cannot determine which other node on a path is chosen to generate a report, but can only determine whether they are chosen or not. A greedy node may use this knowledge to drop all packets but those that contain its own reports. In this case, either the source or destination will change the path because they will only receive reports from a single node on the path.

Consider the cases in which an intermediate node manipulates the reporting node selection or a report to hide its behavior. For instance, the intermediate node may replace the token in a packet with a random value, which results in no report in the packet. Barring a link break, the packet containing the replaced token will be transmitted to the destination. The destination checks packet integrity and decrypts the packet to determine the selected node's identifier. Next, it checks the token integrity and determines that the token was manipulated. Similar to the first case, the destination or source can establish a new path since it determines that one of intermediate nodes on the current path is not trustful.

Let us suppose that in Figure 2, node A forwards a packet and node B drops the packet. In this case, node B may purposefully choose to not send an ACK to node A to hide its malicious behavior. If this occurs, however, node A will send a route error message to the source. The source will change to a new data path.

Instead of withholding ACKs, node B may attempt to hide its selfish behavior by sending an ACK to node A with a random value for the chained HMAC and subsequently dropping the packet. In ASR, the destination can determine if α is incorrect. As shown in Figure 2, the destination retains knowledge of the state by keeping a table consisting of the node identifier and the most recently received chained HMAC. The destination determines the expected α by calculating the chained HMAC. Hence, to protect the flow, either the source or the destination can change to a new path which does not include nodes A and B. Likewise, node A may drop packets and yet generate a report in which it inserts bogus α. This case is similarly detected as the one discussed above. In both cases, however, the destination cannot distinguish whether the fallacious α is generated by A or B, i.e., which node implicates the other. This is a limitation of ASR.

A malicious node can use different techniques than the aforementioned ways, such as dropping packets or manipulating reports, to veil malicious selfish behavior. The node may fabricate a report claiming that it forwarded more packets than it has. For example, in Figure 2, node A may send a report saying it forwarded 100 packets when it really only forwarded 50 packets. In order for A to cheat under the chained HMAC scheme, it must provide α_{100} and β_{100} to the destination and source, respectively. Since the generation of α_{100} and β_{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} and β_{100} is for node B to tell it. This only occurs after A forwards 100 packets to B.

Lastly, nodes which once forwarded packets may attempt to replay reports. Since the report validation scheme uses the sequence number field to compute α_i and β_i, this prevents report replay. The proposed secure random reporting protocol collects reports from intermediate nodes en route, keeping track of the routing path of packets.

B. Nodes in Collusion

Now we discuss collusion attacks by multiple nodes. We first categorize different colluding scenarios based on participating nodes. Let us define an *out-of-path* node to represent a node which is not on a path of a flow and an *in-path* node to

denote a node which is on a path and forwarding traffic for a flow. There exist four different colluding forms: collusion between out-of-path nodes, collusion between out-of-path and in-path nodes, collusion between non-adjacent in-path nodes, and collusion between adjacent in-path nodes.

Out-of-path nodes within transmission range of in-path nodes can eavesdrop traffic flows. These nodes may claim that they have forwarded packets for a flow. In ASR, however, to forge a report, out-of-path nodes need to know the path, selected node, and pairwise keys that the selected node shares with the source and destination. Unless these nodes collaborate with intermediate nodes, this is not possible.

Similarly, suppose that an out-of-path node is within transmission range of intermediate nodes on a path and collaborates with in-path nodes. The out-of-path node may strive to collect information to determine which node is selected by eavesdropping and inform the collaborating in-path node of the information. However, due to the hop-by-hop wrapping scheme in ASR, the out-of-path nodes cannot determine which node has generated a report.

Non-adjacent in-path nodes may forge reports by exchanging information to conceal their selfish behavior. For example, suppose that nodes 1 and 3 in Figure 3 (a) collude and node 1 drops packets. Node 1 may attempt to provide information for node 3 to generate a false report. In this case, however, the colluding in-path nodes are adjacent to cooperative nodes on a path. The colluders do not have the chained HMACs generated by the cooperative nodes adjacent to them since they did not forward packets.

In a similar and more powerful way, adjacent nodes, e.g. nodes 2 and 3 in Figure 3 (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 (3 in the example), however, which is adjacent to non-colluding node (node 4), cannot generate a report stating that it forwarded all the packets to node 4. In order to generate this report, the node (node 3) needs α_i that can be generated only by the next hop cooperative node (node 4) when it receives i packets. Hence, the last colluding node will be detected by the forgery detection scheme. This way, nodes in collusion will gradually be excluded from routing. However, identifying all colluding adjacent nodes at once is a challenging and difficult issue. To limit a colluding attack by adjacent nodes, other approaches such as game theoretic schemes [26] may be used such that nodes behave fairly to maximize their utility.

VII. Performance and Overhead Analysis

In order to analyze the reliability of the proposed reporting schemes, we simulated our protocols using ns-2. These simulations illustrate the robustness of three random reporting protocols (RRNS, RRNDS, and RBR). Additionally, we explore the overhead of packet size and memory, and the cost of the underlying cryptographic constructions based on empirical data.

A. Simulation Environment

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 [8] was used to create a network consisting of 50 mobile nodes in an 1500m X 300m range. A random waypoint mobility model with speeds of 20 m/second was used, in which each node moves to random location in the specified network area with an average moving speed which ranges from minimum speed to maximum speed. Once a node locates to the target location, it remains in that position for a time (pause time) before moving to another random location.

The effects of mobility and traffic loads on three secure random reporting protocols are analyzed by running simulations with 4 different pause times. We include 25 and 30 CBR background traffic flows in our simulations. In each case, the target source and destination are randomly selected from 50 nodes.

TABLE II
Simulation Parameters

Simulation Time	900 seconds
Number of nodes	50
Packet Size	512 bytes
Mobility	Random waypoint mobility model (20 meter/second)
Routing Protocol	Dynamic Source Routing (DSR)
Data Rate	4 packets/second
Transport Protocol	UDP

The period in which the destination and source observe the reports is fixed to allow proper analysis. We heuristically determined a basic observation period of ten seconds based on the speed (average 10 m/s) and transmission range (250 m). In order to adapt to the dynamic characteristic of path changes, reports were collected based on the flow and path. Traffic flows are defined by the source and destination addresses. As paths change, the source and destination keep track of the flow state, the current path state, and the active path list consisting of paths transmitting the traffic during the observation period.

When a node encounters a link failure, it sends a Route Error to the source as defined in the DSR routing protocol. Until the source receives the Route Error, it continues to use the current path. Therefore, packets transmitted to this path before the source changes a path may be lost due to a broken link. In the detection algorithm presented in the next subsection, the Route Error message helps the source estimate the number of lost packets due to link breakage, not a malicious node on a path.

B. Simulation Results and Discussion

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. For the flow of the designated source and destination, we implement an adversarial setting in which if a path has more than one intermediate node, one intermediate node on the path is set to behave maliciously by dropping packets. 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 (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. Here, the detection algorithms themselves are not important; they are for illustrative purposes only to quantify the impact of the report protocol variants.

Case 1: The dropping rate of a malicious node is set 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 a different speed; we chose a slightly higher value (17%) than the average (12%) to accommodate the variance of packet loss rate. We designated any node shown to have a loss rate greater than 12% as anomalous, during the observation period (ten seconds).

Figure 4 shows the impact of mobility on the effectiveness of the three protocols under this simple attack. All points are the averaged value over 5 runs. 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 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 know the network statistics and use it to schedule packet drops. For example, an adversary may 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 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 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.

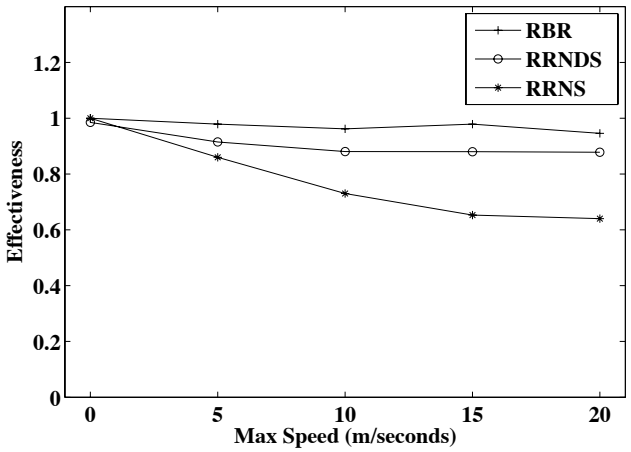
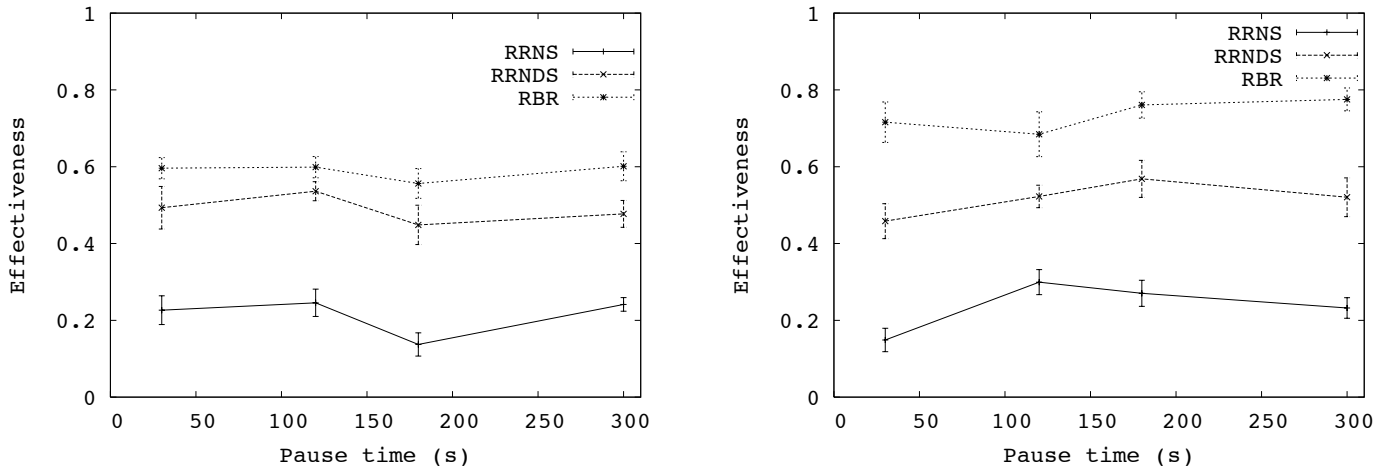


Fig. 4. Effectiveness vs. Mobility in a fixed attacking rate

Figure 5 shows the effectiveness of three random reporting protocols under the intelligent attack strategy, using 25 and 30 CBR background traffic sources respectively. The figure also shows 95% confidence interval and average effectiveness where all points are the averaged value over 10 runs. 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 therefore the 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 is 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 gains useful information from Route Errors, which allows the source to evaluate the number of packets lost due to link failure on the path. Conversely, 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 Route Errors 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 Route Errors. This improves the effectiveness of RRNDS and RRNS by 56% and 280% respectively, but is



(a) 25 CBR Sources (b) 30 CBR Sources

Fig. 5. Effectiveness vs. Mobility/Loads

still lower than RBR's effectiveness. While sending all reports to the source appears to be advantageous, it is vulnerable to a report dropping attack by downstream nodes. When the communication is unidirectional (source-to-destination), the selected node has to generate an extra packet to send a report to the source node. A node downstream 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 in both directions can help the detection algorithms in the source and destination.

Case 3: Finally, we conducted simulations to compare resiliency of ASR with a reference scheme in which intermediate nodes periodically send a separate report of their contribution to a source. Because we do not assume an Internet connection, we have the report sent to the source.

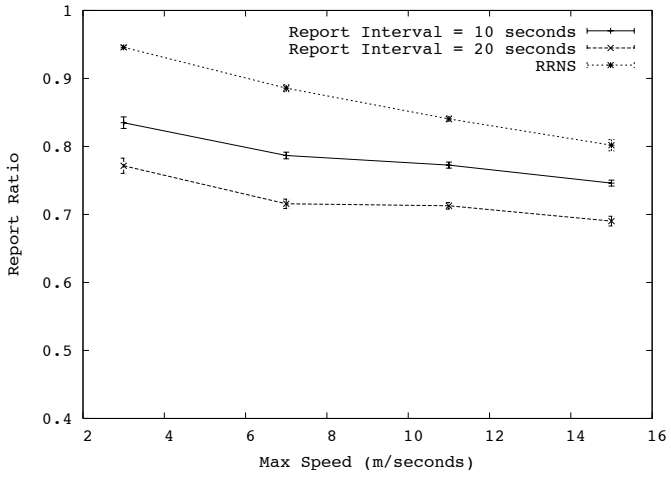


Fig. 6. Reporting Ratio vs. Mobility

In these simulations, there are 10 CBR connections in which each source generates a CBR packet (512 bytes) per second, and mobile nodes move at a speed uniformly distributed from 0 m/s to a maximum without pause. The simulation was run for 500 seconds.

The report ratio is defined as the ratio of total reported number of forwarded packets to total number of packets forwarded by intermediate nodes. We compared the reference scheme with RRNS which is the least robust version from three random reporting protocols. Figure 6 shows the simulation result of the averaged report ratio and 95% confidence interval, where all points are the averaged value over 10 runs. As the report interval becomes short, the amount of reported information in the reference scheme gets close to RRNS, while frequent report transmission incurs communication overhead. From this simulation, the piggybacking of a report in ASR with the random node selection provides an efficient and effective solution to collect reports.

C. Overhead

In MANETs, mobile nodes have limited batteries, small memory, and limited computational power. Therefore, it is imperative to analyze the overhead caused by ASR. In this subsection, we explore the overhead of ASR: packet size, memory overhead, and computational overhead.

Packet Size: The secure random reporting protocol requires two new fields: $Report$ and $Token$. The $Report$ field consists of two reports: forward report (R_f) and backward report (R_b). Each report is composed of an initial sequence number (4 bytes), a chained HMAC (16 bytes for MD5), a sequence number map (*smap* 4 bytes), and the number of packets (4 bytes) that the selected node forwarded. The $Token$ is a cryptographic hash output (16 bytes for MD5). The total overhead incurred by the secure random reporting protocol is 68 bytes which is only 4.5% of maximum data packet size. On the other hand, existing eavesdropping schemes require a separate packet transmission of the report to other specific nodes or a centralized server. This separate packet transmission incurs additional transmission delay and energy consumption. Our reporting protocol removes these extra costs by embedding a report in data packet with small overhead in packet size.

Memory Overhead: In ASR, each intermediate node on the path maintains only current state information of a specific flow. The state consists of path information (20 bytes if a path has 5 nodes), sequence number map (4 bytes), the number of forwarded packets (4 bytes), α and β (32 bytes). Here, the path information is used for both the forward and backward traffic, while other state information should be maintained separately for the forward and backward flows, which requires 100 bytes in total. The source and destination should keep the information of intermediate nodes on a path which consists of sequence number, the number of forwarded packets, and α and β during a monitoring period. The monitoring period is decided by the source and destination. Let us assume that the source generates one packet per second, the monitoring period is 10 seconds, and a source route has five intermediate nodes. In this setting, the source and destination only need about 400 bytes, respectively. The chained HMAC scheme enables the source and destination to keep the information of each intermediate node, and intermediate nodes to maintain current state information of a flow, independent of the number of received packets.

Computation Overhead: One 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). Results were averaged using the raw cycle count over 1000 test runs to ensure accurate results.

In our simulation, the longest path has ten intermediate nodes. For this 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 in which 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 results in $55.81\ \mu s$ overhead.

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

Algorithm	HMAC Chain	HMAC Data	Total
MD5	7.037 μs	29.950 μs	271.01 μs
SHA1	19.108 μs	86.614 μs	746.468 μs
SHA256	20.308 μs	95.606 μs	800.452 μs

The total computational cost consists of four factors: two HMACs of data packets at the source and destination, two chained HMACs (α_i and β_i) 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 μs overhead.

To measure end-to-end transmission delay, we set a condition in which packets does not experience link failure and congestion. In this optimal setting, data transmission from the source to the destination takes 65 milliseconds on a path which has 10 intermediate nodes (6 milliseconds per hop transmission). The total HMAC, hash, and AES computation takes only 1.3% of the total time.

If we map the results to a PDA, we still find that the computational overhead is low. For example, PDA processor speeds range from 200 MHz to over 600 MHz. For the 200 MHz PDA, the clock cycles are four times longer than our laptop. Suppose that we choose to use SHA256. The total computation of ASR in a 200 MHz PDA will take approximately 3.42 milliseconds.

VIII. DISCUSSION

In this section we discuss the trade-offs of ASR in the presence of unidirectional traffic flows. Although most application protocols are bidirectional, there exist several unidirectional protocols, e.g. multimedia services using UDP. If the traffic flow is unidirectional, the selected node cannot attach self-report to the reverse directional data packet. To decide if the traffic flow is unidirectional, the intermediate node may inspect packet headers, or use timeouts to detect source-bound packets. Once the intermediate node detects the unidirectional flow, it has two choices: either it generates a separate report or does not send a report towards the source.

With the first choice, the selected node sends a separate packet carrying a report to the source. As simulation results in Section VII show, this bidirectional reporting improves the robustness of reports. However, the selected node's neighbors may eavesdrop on the communication and determine that the selected node has generated a report. By choosing this option, the node selection will not be anonymous. Moreover, the separate report packet incurs communication overhead.

With the second choice, the intermediate node does not generate a separate report towards the source, which becomes RRNS. This reduces the robustness of reports as shown by the simulation results. However, this way the selection of the reporting node remains anonymous to neighbors, without incurring extra communication overhead.

IX. Conclusions

Most military applications of MANETs target mission oriented scenarios such as battlefields 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.

X. Acknowledgments

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

References

[1] L. Anderegg and S. Eidenbenz. Ad hoc-VCG: a Truthful and Cost-Efficient Routing Protocol for Mobile Ad Hoc NetworksWith Selfish Agents. *AC Mobicom*, 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] L. Buttyan and J.-P. Hubaux. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. *Mobile Networks and Applications*, 2003.

[5] H. Chan and a. S. A. Perrig. Random key predistribution schemes for sensor networks. *In Proceedings of the IEEE Symposiumon Security and Privacy*, 2003.

[6] D. L. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonym s. *Communications of the ACM*, 1981.

[7] L. Eschenauer and V. Gligor. A key management scheme for distributed sensor networks. *In Proceedings of ACM Conference on Computer and Communications Security*, 2002.

[8] http://www.isi.edu. The Network Simulator - ns-2, 2000.

[9] Y.-C. Hu and A. Perrig. A Survey of Secure Wireless Ad Hoc Routing. *IEEE Security and Privacy, special issue on Making Wireless Work*, 2004.

[10] Y. 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.

[11] 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.

[12] 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.

[13] J. Kong and X. Hong. ANODR:ANonymous On Demand Routing with Untraceable Routes for Mobile Ad-hoc Networks. *In ACM MOBIHOC*, 2003.

[14] D. Liu and P. Neng. Establishing pairwise keys in distributed sensor networks. *In Proceedings of ACM Conference on Computer and Communications Security*, 2003.

[15] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. *Proc. of ACM Mobicom*, 2000.

[16] S. Micali and R. Rivest. Micropayments Revisited. *CT-RSA*, 2002.

[17] 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.

[18] C. E. Perkins and E. Belding-Royer. Ad hoc On-Demand Distance Vector (AODV) Routing. *IETF RFC3561*, 2003.

[19] 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.

[20] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[21] V. Srinivasan, P. Nuggehalli, C. F. Chiasserini, and R. R. Rao. Cooperation in Wireless Ad Hoc Networks. *IEEE INFOCOM*, 2003.

[22] 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.

[23] Y. Zhang and W. Lee. Intrusion Detection in Wireless Ad Hoc Networks. *6th International Conference Mobile Computing and Networks*, 2000.

[24] Y. Zhang and W. Lee. Intrusion Detection in Wireless Ad Hoc Networks. *Proc. of ACM Mobicom*, 2000.

[25] Y. Zhang, W. Liu, and W. Lou. Anonymous Communications in Mobile Ad Hoc Networks. *IEEE INFOCOM*, 2005.

[26] S. Zhone, J. Chen, and Y. R. Yang. Sprite: A Simpe, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks. *Proc. of IEEE INFOCOM*, 2003.

[27] 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.

[28] S. Zhu, S. Xu, S. Setia, and S. Jajodia. Establishing Pair-wise Keys For Secure Communication in Ad Hoc Networks: A Probabilistic Approach. *IEEE International Conference on Network Protocols (ICNP'03)*, 2003.

[29] 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.