

# Detection of Malicious Packet Dropping in Wireless Ad Hoc Networks Based on Privacy-Preserving Public Auditing

Tao Shu  
Department of Computer Science and  
Engineering  
Oakland University  
Rochester, MI 48309, USA  
shu@oakland.edu

Marwan Krunz  
Department of Electrical and Computer  
Engineering  
University of Arizona  
Tucson, AZ 85721, USA  
krunz@email.arizona.edu

## ABSTRACT

In a multi-hop wireless ad hoc network, packet losses are attributed to harsh channel conditions and intentional packet discard by malicious nodes. In this paper, while observing a sequence of packet losses, we are interested in determining whether losses are due to link errors only, or due to the combined effect of link errors and malicious drop. We are especially interested in insider's attacks, whereby a malicious node that is part of the route exploits its knowledge of the communication context to selectively drop a small number of packets that are critical to network performance. Because the packet dropping rate in this case is comparable to the channel error rate, conventional algorithms that are based on detecting the packet loss rate cannot achieve satisfactory detection accuracy. To improve the detection accuracy, we propose to exploit the correlations between lost packets. Furthermore, to ensure truthful calculation of these correlations, we develop a homomorphic linear authenticator (HLA) based public auditing architecture that allows the detector to verify the truthfulness of the packet loss information reported by nodes. This architecture is privacy preserving, collusion proof, and incurs low communication and storage overheads. Through extensive simulations, we verify that the proposed mechanism achieves significantly better detection accuracy than conventional methods such as a maximum-likelihood based detection.

## Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: General—Security and Protection (*e.g.*, firewalls)

## General Terms

Security, reliability, algorithms, design

## Keywords

Denial-of-service, malicious user detection, homomorphic linear authentication, wireless ad hoc networks, security

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WiSec'12, April 16–18, 2012, Tucson, Arizona, USA.

Copyright 2012 ACM 978-1-4503-1265-3/12/04 ...\$10.00.

## 1. INTRODUCTION

### 1.1 Motivation

In a multi-hop wireless network, nodes cooperate in relaying/routing traffic. An adversary can exploit this cooperative nature to launch denial-of-service (DoS) attacks. For example, the adversary may first pretend to be a cooperative node in the route discovery process. Once being included in a route, the adversary may start maliciously dropping packets. In the most straightforward form of this attack, the malicious node simply stops forwarding packets received from upstream nodes, completely disrupting the traffic delivery between the source and the destination. Eventually, such severe DoS attacks can paralyze the network by partitioning its topology.

Even though persistent packet dropping can effectively degrade the performance of the network, from the attacker's standpoint performing such an "always-on" attack has its disadvantages in terms of the ease of detection [22]. A malicious node that is part of the route can actually exploit its knowledge of the network protocols and the communication context to launch an *insider's attack*, aiming at achieving the same attack effect but at a much lower risk of being detected. Specifically, the malicious node can identify the importance of various packets and drop a small number of packets that are deemed highly critical to the performance of the network. These important packets are typically control packets. For example, in a frequency-hopping network, these packets may convey frequency hopping sequences; in an ad hoc cognitive radio network, they could be the packets that carry the idle channel lists (*i.e.*, white spaces) that are used to establish a network-wide control channel. By targeting these critical packets, the authors in [18, 21, 22] have shown that a non-persistent insider's attack can cause significant damage to the network performance. In this paper, we are interested in combating such an insider's attack. In particular, we are interested in the problem of detecting the events of selective packet drops and identifying the malicious node(s) responsible for these drops.

Detecting malicious selective packet dropping is extremely challenging in a highly dynamic wireless environment. The difficulty stems from the requirement that we need to not only detect the location (or hop) where the packet drop took place, but also identify whether the drop is intentional or not. Specifically, because of the open nature of the wireless medium, the quality of the channel typically fluctuates due to fading, shadowing, interference, and background noise. As a result, a packet drop in the route could be caused by harsh channel conditions (*a.k.a.*, link errors) or by malicious

behavior. In some cases, e.g., a highly mobile environment, link errors are quite significant. So, a malicious node can camouflage its attack under the background of harsh channel conditions by selectively dropping a small number of highly important packets. In this case, observing the packet loss rate is not enough to accurately identify the exact cause of a packet loss, because the packet drop rate by the malicious node is comparable to that of wireless link errors. Clearly, deciding whether a packet drop is intentional or unintentional in such an ambiguous setup is a challenging problem.

The above problem has not been well addressed in the literature. As discussed in Section 2, most of the related works preclude the ambiguity of the environment by assuming that malicious dropping is the only source of packet loss, so that there is no need to account for the impact of link errors. On the other hand, for the small number of works that differentiate between link errors and malicious packet drops, their detection algorithms usually require the number of dropped packets by the attacker to be significantly higher than link errors, in order to provide an acceptable detection accuracy.

## 1.2 Main Contribution and Paper Organization

In this paper, we develop an accurate algorithm for detecting selective packet drops made by insider malicious nodes. Our algorithm also provides a truthful and publicly verifiable decision statistics as a proof to support the detection decision. The high detection accuracy is achieved by exploiting the correlations between the positions of lost packets, as calculated from the packet-loss bitmap (a bitmap describing the lost/received status of each packet in a sequence of consecutive packet transmissions). The basic idea behind this method is that even though malicious dropping may result in a packet loss rate that is comparable to normal channel losses, the stochastic processes that characterize the two phenomena exhibit different correlation structures (equivalently, different patterns of packet losses). Therefore, by detecting the correlations between lost packets, one can decide whether the packet loss is purely due to regular link errors, or is a combined effect of link error and malicious drop. Our algorithm takes into account the cross-statistics between lost packets to make a more informative decision, and thus is in sharp contrast to the conventional methods that rely only on the distribution of the number of lost packets.

The main challenge in realizing our mechanism lies in how to guarantee that the packet-loss bitmaps reported by individual nodes along the route are truthful, i.e., reflect the actual status of each packet transmission. Such truthfulness is essential for correct calculation of the correlation between lost packets. This challenge is not trivial, because it is natural for an attacker to report false information to the detection algorithm to avoid being detected. For example, the malicious node may understate its packet-loss bitmap, i.e., some packets may have been dropped by the node but the node reports that these packets have been forwarded. Therefore, some auditing mechanism is needed to verify the truthfulness of the reported information. Considering that a typical wireless device is resource-constrained, we also require that a user should be able to delegate the burden of auditing and detection to some public server to save its own resources.

Our solution to the above public-auditing problem is constructed based on the homomorphic linear authenticator (HLA) cryptographic primitive [2][3][24], which is basically a signature scheme widely used in cloud computing and storage

server systems to provide a proof of storage from the server to entrusting clients [25]. However, direct application of HLA does not solve our problem well, mainly because in our problem setup, there can be more than one malicious node along the route. These nodes may collude (by exchanging information) during the attack and when being asked to submit their reports. For example, a packet and its associated HLA signature may be dropped at an upstream malicious node, so a downstream malicious node does not receive this packet and the HLA signature from the route. However, this downstream attacker can still open a back-channel to request this information from the upstream malicious node. When being audited, the downstream malicious node can still provide valid proof for the reception of the packet. So packet dropping at the upstream malicious node is not detected. Such collusion is unique to our problem, because in the cloud computing/storage server scenario, a file is uniquely stored at a single server, so there are no other parties for the server to collude with. We show that our new HLA construction is collusion-proof.

Our construction also provides the following new features. First, *privacy-preserving*: the public auditor should not be able to discern the content of a packet delivered along the route through the auditing information submitted by individual hops, no matter how many independent reports of the auditing information are submitted to the auditor. Second, our construction incurs low communication and storage overheads at intermediate nodes. This makes our mechanism applicable to a wide range of wireless devices, including low-cost wireless sensors that have very limited bandwidth and memory capacities. This is also in sharp contrast to the typical storage-server scenario, where bandwidth/storage is not considered an issue.

The remainder of this paper is organized as follows. In Section 2 we review the related work. The system/adversary models and problem statement are described in Section 3. We present the proposed scheme and analyze its security performance and overheads in Section 4. Simulation results are presented in Section 5, and we conclude the paper in Section 6.

## 2. RELATED WORK

Depending on how much weight a detection algorithm gives to link errors relative to malicious packet drops, the related work can be classified into the following two categories.

The first category aims at high malicious dropping rates, where most (or all) lost packets are caused by malicious dropping. In this case, the impact of link errors is ignored. Most related work falls into this category. Based on the methodology used to identify the attacking nodes, these works can be further classified into four sub-categories. The first sub-category is based on credit systems [7][27]. A credit system provides an incentive for cooperation. A node receives credit by relaying packets for others, and uses its credit to send its own packets. As a result, a maliciously node that continuously to drop packets will eventually deplete its credit, and will not be able to send its own traffic. The second sub-category is based on reputation systems [9][6][11][16][17][8][4]. A reputation system relies on neighbors to monitor and identify misbehaving nodes. A node with a high packet dropping rate is given a bad reputation by its neighbors. This reputation information is propagated periodically throughout the network and is used as an important metric in selecting routes. Consequently, a

malicious node will be excluded from any route. The third sub-category of works relies on end-to-end or hop-to-hop acknowledgements to directly locate the hops where packets are lost [15][19][20]. A hop of high packet loss rate will be excluded from the route. The fourth sub-category addresses the problem using cryptographic methods. For example, the work in [14] utilizes Bloom filters to construct proofs for the number of packets that are forwarded at each node. By examining the number of relayed packets at successive hops along a route, one can identify suspicious hops that exhibit high packet loss rates. Similarly, the method in [13] traces the forwarding records of a particular packet at each intermediate node by formulating the tracing problem as a Renyi-Ulam game. The first hop where the packet is no longer forwarded is considered a suspect for misbehaving.

The second category targets the scenario where the number of maliciously dropped packets is significantly higher than that caused by link errors, but the impact of link errors is non-negligible. Certain knowledge of the wireless channel is necessary in this case. The authors in [23] proposed to shape the traffic at the MAC layer of the source node according to a certain statistical distribution, so that intermediate nodes are able to estimate the rate of received traffic by sampling the packet arrival times. By comparing the source traffic rate with the estimated received rate, the detection algorithm decides whether the discrepancy in rates, if any, is within a reasonable range such that the difference can be considered as being caused by normal channel impairments only, or caused by malicious dropping, otherwise. The works in [10] and [26] proposed to detect malicious packet dropping by counting the number of lost packets. If the number of lost packets is significantly larger than the expected packet loss rate made by link errors, then with high probability a malicious node is contributing to packet losses.

All methods mentioned above do not perform well when malicious packet dropping is highly selective. More specifically, for the credit-system-based method, a malicious node may still receive enough credits by forwarding most of the packets it receives from upstream nodes. Similarly, in the reputation-based approach, the malicious node can maintain a reasonably good reputation by forwarding most of the packets to the next hop. As for the acknowledgement-based method, the Bloom-filter scheme, and all the mechanisms in the second category, merely counting the number of lost packets does not give a sufficient ground to detect the real culprit that is causing packet losses. This is because the difference in the number of lost packets between the link-error-only case and the link-error-plus-malicious-dropping case is small when the attacker drops only a few packets. Consequently, the detection accuracy of these algorithms deteriorates when malicious drops become highly selective.

Our study targets the challenging situation where link errors and malicious dropping lead to comparable packet loss rates. The effort in the literature on this problem has been quite preliminary, and there is a few related works. Note that the cryptographic methods proposed in [21] to counter selective packet jamming target a different issue than the detection problem studied in this paper. The methods in [21] delay a jammer from recognizing the significance of a packet after the packet has been successfully transmitted, so that there is no time for the jammer to conduct jamming based on the content/importance of the packet. Instead of trying to detect any malicious behavior, the approach in [21] is proactive, and hence incurs overheads regardless of the presence or absence of attackers.

### 3. SYSTEM MODELS AND PROBLEM STATEMENT

#### 3.1 Network and Channel Models

Consider an arbitrary path  $P_{SD}$  in a multi-hop wireless ad hoc network, as shown in Figure 1. The source node  $S$  continuously sends packets to the destination node  $D$  through intermediate nodes  $n_1, \dots, n_K$ , where  $n_i$  is the upstream node of  $n_{i+1}$ , for  $1 \leq i \leq K-1$ . We assume that  $S$  is aware of the route  $P_{SD}$ , as in Dynamic Source Routing (DSR) [12]. If DSR is not used,  $S$  can identify the nodes in  $P_{SD}$  by performing a traceroute operation.

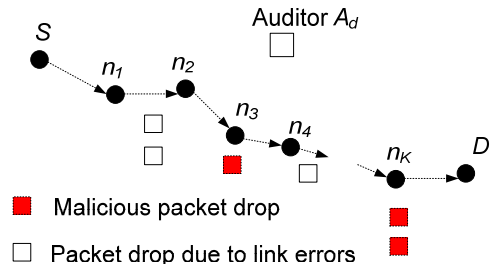


Figure 1: Network and attack model.

We model the wireless channel of each hop along  $P_{SD}$  as a random process that alternates between good and bad states. Packets transmitted during the good state are successful, and packets transmitted during the bad state are lost. In contrast to the classical Gilbert-Ellioit (GE) channel model, here we do not assume any Markovian property on the channel behavior. We only require that the sequence of sojourn times for each state follows a stationary distribution, and the autocorrelation function of the channel state, say  $f_c(i)$ , where  $i$  is a discrete time lag measured in packets, is also stationary. The function  $f_c(i)$  can be calculated using the probing approach in [1]. In brief, a sequence of  $M$  packets are transmitted consecutively over the channel. By observing whether the transmissions are successful or not, the receiver obtains a realization of the channel state  $(a_1, \dots, a_M)$ , where  $a_j \in \{0, 1\}$  for  $j = 1, \dots, M$ . In this sequence, “1” denotes the packet was successfully received, and “0” denotes the packet was dropped.  $f_c(i)$  is derived by computing the auto-correlation function of this sample sequence. Such measurement can take place online or offline. A detailed discussion on how  $f_c(i)$  is derived is out of the scope of this paper, and we simply assume that this information is given as input to our detection algorithm.

There is an independent auditor  $A_d$  in the network.  $A_d$  is independent in the sense that it is not associated with any node in  $P_{SD}$  and does not have any knowledge of the secrets (e.g., cryptographic keys) held by various nodes. The auditor is responsible for detecting malicious nodes on demand. Specifically, we assume  $S$  receives feedback from  $D$  when  $D$  suspects that the route is under attack. Such a suspicion may be triggered by observing any abnormal events, e.g., a significant performance drop, the loss of multiple packets of a certain type, etc. We assume that the integrity and authenticity of the feedback from  $D$  to  $S$  can be verified by  $S$  using resource-efficient cryptographic methods such as the Elliptic Curve Digital Signature Algorithm (ECDSA). Once being notified of possible attacks,  $S$  submits an *attack-detection request* (ADR) to  $A_d$ . To facilitate its investigation,  $A_d$  needs to collect certain information (elaborated on

in the next section) from the nodes on route  $P_{SD}$ . We assume that each such node must reply to  $A_d$ 's inquiry, otherwise the node will be considered as misbehaving. We assume that normal nodes will reply with truthful information, but malicious nodes may cheat. At the same time, for privacy reasons, we require that  $A_d$  cannot determine the content of the normal packets delivered over  $P_{SD}$  from the information collected during the auditing.

### 3.2 Adversarial Model

The goal of the adversary is to degrade the network's performance by maliciously dropping packets while remaining undetected. We assume that the malicious node has knowledge of the wireless channel, and is aware of the algorithm used for misbehavior detection. It has the freedom to choose what packets to drop. For example, in the random-drop mode, the malicious node may drop any packet with a small probability  $p_d$ . In the selective-mode, the malicious node only drops packets of certain types. A combination of the two modes may be used. We assume that any node on  $P_{SD}$  can be a malicious node, except the source and the destination. In particular, there can be multiple malicious nodes on  $P_{SD}$ .

We consider the following form of collusion between malicious nodes: A covert communication channel may exist between any two malicious nodes, in addition to the path connecting them on  $P_{SD}$ . As a result, malicious nodes can exchange any information without being detected by  $A_d$  or any other nodes in  $P_{SD}$ . Malicious nodes can take advantage of this covert channel to hide their misbehavior and reduce the chance of being detected. For example, an upstream malicious node may drop a packet on  $P_{SD}$ , but may secretly send this packet to a downstream malicious node via the covert channel. When being investigated, the downstream malicious node can provide a proof of the successful reception of the packet. This makes the auditor believe that the packet was successfully forwarded to the downstream nodes, and not know that the packet was actually dropped by an upstream attacker.

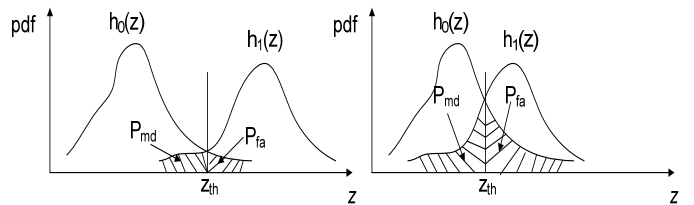
### 3.3 Problem Statement

Under the system and adversary models defined above, we address the problem of identifying the nodes on  $P_{SD}$  that drop packets maliciously. We require the detection to be performed by a public auditor that does not have knowledge of the secrets held by the nodes on  $P_{SD}$ . When a malicious node is identified, the auditor should be able to construct a publicly verifiable proof of the misbehavior of that node. The construction of such a proof should be privacy preserving, i.e., it does not reveal the original information that is transmitted on  $P_{SD}$ . In addition, the detection mechanism should incur low communication and storage overheads, so that it can be applied to a wide variety of wireless networks.

## 4. PROPOSED DETECTION SCHEME

### 4.1 Overview

The main idea of our detection algorithm is to compare the autocorrelation function of the observed packet loss process of a link with that of a normal wireless channel (i.e.,  $f_c(i)$ ) to accurately identify any possible malicious packet drops. The necessity of exploiting the correlation of lost packets to improve the detection accuracy can be illustrated by examining the insufficiency of the conventional method that relies only on the distribution of the number of lost



(a) mean of  $y$  much greater than mean of  $x$  (b) mean of  $y$  is comparable to mean of  $x$

**Figure 2: Insufficiency of conventional detection algorithms when malicious packet drops are highly selective.**

packets. More specifically, under the conventional method, malicious-node detection is modeled as a binary hypothesis test, where  $H_0$  is the hypothesis that there is no malicious node in a given link (all packet losses are due to link errors) and  $H_1$  denotes there is a malicious node in the given link (packet losses are due to both link errors and malicious drops). Let  $z$  be the observed number of lost packets on the link during some interval  $t$ . Then,

$$z = \begin{cases} x, & \text{under } H_0 \text{ (no malicious nodes)} \\ x + y, & \text{under } H_1 \text{ (there is a malicious node)} \end{cases} \quad (1)$$

where  $x$  and  $y$  are the numbers of lost packets caused by link errors and by malicious drops, respectively. Both  $x$  and  $y$  are random variables. Let the probability density functions of  $z$  conditioned on  $H_0$  and on  $H_1$  be  $h_0(z)$  and  $h_1(z)$ , respectively, as shown in Figure 2(a). We are interested in the maximum-uncertainty scenario where the *a priori* probabilities are given by  $\Pr\{H_0\} = \Pr\{H_1\} = 0.5$ , i.e., the auditor has no prior knowledge of the distributions of  $H_0$  and  $H_1$  to make any biased decision regarding the presence of malicious nodes. Let the false-alarm and miss-detection probabilities be  $P_{fa}$  and  $P_{md}$ , respectively. The optimal decision strategy that minimizes the total detection error  $P_{de} \stackrel{\text{def}}{=} 0.5(P_{fa} + P_{md})$  is the maximum-likelihood (ML) algorithm:

$$\begin{cases} \text{if } z \leq z_{th}, & \text{accept } H_0 \\ \text{otherwise,} & \text{accept } H_1 \end{cases} \quad (2)$$

where the threshold  $z_{th}$  is the solution to the equation  $h_0(z_{th}) = h_1(z_{th})$ . Under this strategy,  $P_{fa}$  and  $P_{md}$  are the areas of the shaded regions shown in Figure 2(a), respectively. The problem with this mechanism is that, when the mean of  $y$  is small,  $h_1(z)$  and  $h_0(z)$  are not sufficiently separated, leading to large  $P_{fa}$  and  $P_{md}$ , as shown in Figure 2(b). This observation implies that when malicious packet drops are highly selective, counting the number of lost packets is not sufficient to accurately differentiate between malicious drops and link errors. For such a case, we use the correlation between lost packets to form a more solid decision statistic.

To correctly calculate the correlation between lost packets, it is critical to enforce a truthful packet-loss bitmap report by each node. We use HLA cryptographic primitive for this purpose. The basic idea of our method is as follows. An HLA scheme allows the source, which has knowledge of the HLA secret key, to generate HLA signatures  $s_1, \dots, s_M$  for  $M$  independent messages  $r_1, \dots, r_M$ , respectively. The source sends out the  $r_i$ 's and  $s_i$ 's along the route. The HLA signatures are made in such a way that they can be used as the basis to construct a valid HLA signature for any arbitrary linear combination of the messages,  $\sum_{i=1}^M c_i r_i$ , without

the use of the HLA secret key, where  $c_i$ 's are randomly chosen coefficients. A valid HLA signature for  $\sum_{i=1}^M c_i r_i$  can be constructed by a node that does not have knowledge of the secret HLA key if and only if the node has full knowledge of  $s_1, \dots, s_M$ . So, if a node with no knowledge of the HLA secret key provides a valid signature for  $\sum_{i=1}^M c_i r_i$ , it implies that this node must have received all the signatures  $s_1, \dots, s_M$ . Our construction ensures that  $s_i$  and  $r_i$  are sent together along the route, so that knowledge of  $s_1, \dots, s_M$  also proves that the node must have received  $r_1, \dots, r_M$ .

Our detection architecture consists of four phases: setup, packet transmission, audit, and detection. We elaborate on these phases in the next section.

## 4.2 Scheme Details

### 4.2.1 Setup Phase

This phase takes place right after route  $P_{SD}$  is established, but before any data packets are transmitted over the route. In this phase,  $S$  decides on a symmetric-key crypto-system ( $encrypt_{key}$ ,  $decrypt_{key}$ ) and  $K$  symmetric keys  $key_1, \dots, key_K$ , where  $encrypt_{key}$  and  $decrypt_{key}$  are the keyed encryption and decryption functions, respectively.  $S$  securely distributes  $decrypt_{key}$  and a symmetric key  $key_j$  to node  $n_j$  on  $P_{SD}$ , for  $j = 1, \dots, K$ . Key distribution may be based on the public-key crypto-system such as RSA:  $S$  encrypts  $key_j$  using the public key of node  $n_j$  and sends the cipher text to  $n_j$ .  $n_j$  decrypts the cipher text using its private key to obtain  $key_j$ .  $S$  also announces two hash functions,  $H_1$  and  $H_{key}^{MAC}$ , to all nodes in  $P_{SD}$ .  $H_1$  is unkeyed while  $H_{key}^{MAC}$  is a keyed hash function that will be used for message authentication purposes later on.

Besides symmetric key distribution,  $S$  also needs to set up its HLA keys. Let  $e : G \times G \rightarrow G_T$  be a computable bilinear map with multiplicative cyclic group  $G$  and support  $\mathbf{Z}_p$ , where  $p$  is the prime order of  $G$ , i.e., for all  $\alpha, \beta \in G$  and  $q_1, q_2 \in \mathbf{Z}_p$ ,  $e(\alpha^{q_1}, \beta^{q_2}) = e(\alpha, \beta)^{q_1 q_2}$ . Let  $g$  be a generator of  $G$ .  $H_2(\cdot)$  is a secure map-to-point hash function:  $\{0, 1\}^* \rightarrow G$ , which maps strings uniformly to  $G$ .  $S$  chooses a random number  $x \in \mathbf{Z}_p$  and computes  $v = g^x$ . Let  $u$  be another generator of  $G$ . The secret HLA key is  $sk = x$  and the public HLA key is a tuple  $pk = (v, g, u)$ .

### 4.2.2 Packet Transmission Phase

After completing the setup phase,  $S$  enters the packet transmission phase.  $S$  transmits packets to  $P_{SD}$  according to the following steps.

Before sending out a packet  $P_i$ , where  $i$  is a sequence number that uniquely identifies  $P_i$ ,  $S$  computes  $r_i = H_1(P_i)$  and generates the HLA signatures of  $r_i$  for node  $n_j$ , as follows

$$s_{ji} = [H_2(i||j)u^{r_i}]^x, \quad \text{for } j = 1, \dots, K \quad (3)$$

where  $||$  denotes concatenation. These signatures are then sent together with  $P_i$  to the route by using a one-way chained encryption that prevents an upstream node from deciphering the signatures intended for downstream nodes. More specifically, after getting  $s_{ji}$  for  $j = 1, \dots, K$ ,  $S$  iteratively

computes the following:

$$\begin{aligned} \tilde{s}_{Ki} &= encrypt_{key_K}(s_{Ki}) \\ \tau_{Ki} &= \tilde{s}_{Ki} || MAC_{key_K}(\tilde{s}_{Ki}) \\ \tilde{s}_{K-1i} &= encrypt_{key_{K-1}}(s_{K-1i} || \tau_{Ki}) \\ \tau_{K-1i} &= \tilde{s}_{K-1i} || MAC_{key_{K-1}}(\tilde{s}_{K-1i}) \\ &\vdots \\ \tilde{s}_{ji} &= encrypt_{key_j}(s_{ji} || \tau_{j+1i}) \\ \tau_{ji} &= \tilde{s}_{ji} || MAC_{key_j}(\tilde{s}_{ji}) \\ &\vdots \\ \tilde{s}_{1i} &= encrypt_{key_1}(s_{1i} || \tau_{2i}) \\ \tau_{1i} &= \tilde{s}_{1i} || MAC_{key_1}(\tilde{s}_{1i}) \end{aligned} \quad (4)$$

where the message authentication code (MAC) in each stage  $j$  is computed according to the hash function  $H_{key_j}^{MAC}$ . After getting  $\tau_{1i}$ ,  $S$  puts  $P_i || \tau_{1i}$  into one packet and sends it to node  $n_1$ .

When node  $n_1$  receives the packet from  $S$ , it extracts  $P_i$ ,  $\tilde{s}_{1i}$ , and  $MAC_{key_1}(\tilde{s}_{1i})$  from the received packet. Then,  $n_1$  verifies the integrity of  $\tilde{s}_{1i}$  by testing the following equality:

$$MAC_{key_1}(\tilde{s}_{1i}) = H_{key_1}^{MAC}(\tilde{s}_{1i}). \quad (5)$$

If the test is true, then  $n_1$  decrypts  $\tilde{s}_{1i}$  as follows:

$$decrypt_{key_1}(\tilde{s}_{1i}) = s_{1i} || \tau_{2i}. \quad (6)$$

Then,  $n_1$  extracts  $s_{1i}$  and  $\tau_{2i}$  from the decrypted text. It stores  $r_i = H_1(P_i)$  and  $s_{1i}$  in its proof-of-reception database for future use. This database is maintained at every node on  $P_{SD}$ . It can be considered as a FIFO queue of size  $M$ , which records the reception status for the most recent  $M$  packets sent by  $S$ . Finally,  $n_1$  assembles  $P_i || \tau_{2i}$  into one packet and relays this packet to node  $n_2$ . In case the test in (5) fails,  $n_1$  marks the loss of  $P_i$  in its proof-of-reception database and does not relay the packet to  $n_2$ .

The above process is repeated at every intermediate node  $n_j$ ,  $j = 1, \dots, K$ . As a result, node  $n_j$  obtains  $r_i$  and its HLA signature  $s_{ji}$  for every packet  $P_i$  that the node has received, and it relays  $P_i || \tau_{j+1i}$  to the next hop on the route. The last hop, i.e., node  $n_K$ , only forwards  $P_i$  to the destination  $D$ . As proved in Theorem 4 in Section 4.3, the special structure of the one-way chained encryption construction in (4) dictates that an upstream node on the route cannot get a copy of the HLA signature intended for a downstream node, and thus the construction is resilient to the collusion model defined in Section 3.2. Note that here we consider the verification of the integrity of  $P_i$  as an orthogonal problem to that of verifying the tag  $\tau_{ji}$ . If the verification of  $P_i$  fails, node  $n_1$  should also stop forwarding the packet and should mark it accordingly in its proof-of-reception database.

### 4.2.3 Audit Phase

This phase is triggered when the public auditor  $A_d$  receives an ADR message from  $S$ . The ADR message includes the id of the nodes on  $P_{SD}$ , ordered in the downstream direction, i.e.,  $n_1, \dots, n_K$ ,  $S$ 's HLA public key information  $pk = (v, g, u)$ , the sequence numbers of the most recent  $M$  packets sent by  $S$ , and the sequence numbers of the subset of these  $M$  packets that were received by  $D$ . Recall that we assume the information sent by  $S$  and  $D$  is truthful, because detecting attacks is in their interest.  $A_d$  conducts the auditing process as follows.

$A_d$  submits a random challenge vector  $\vec{c}_j = (c_{j1}, \dots, c_{jM})$  to node  $n_j$ ,  $j = 1, \dots, K$ , where the elements  $c_{ji}$ 's are randomly chosen from  $\mathbf{Z}_p$ . Without loss of generality, let the sequence number of the packets recorded in the current proof-of-reception database be  $P_1, \dots, P_M$ , with  $P_M$  being the most recent packet sent by  $S$ . Based on the information in this database, node  $n_j$  generates a packet-reception bitmap  $\vec{b}_j = (b_{j1}, \dots, b_{jM})$ , where  $b_{ji} = 1$  if  $P_i$  has been received by  $n_j$ , and  $b_{ji} = 0$  otherwise. Node  $n_j$  then calculates the linear combination  $r^{(j)} = \sum_{i=1, b_{ji} \neq 0}^M c_{ji} r_i$  and the HLA signature for the combination as follows:

$$s^{(j)} = \prod_{i=1, b_{ji} \neq 0} s_{ji}^{c_{ji}}. \quad (7)$$

Node  $n_j$  submits  $\vec{b}_j$ ,  $r^{(j)}$ , and  $s^{(j)}$  to  $A_d$ , as proof of the packets it has received.

$A_d$  checks the validity of  $r^{(j)}$  and  $s^{(j)}$  by testing the following equality:

$$e(s^{(j)}, g) = e\left(\prod_{i=1, b_{ji} \neq 0}^M H_2(i|j)^{c_{ji}} u^{r^{(j)}}, v\right). \quad (8)$$

If the equality holds, then  $A_d$  accepts that node  $n_j$  received the packets as reflected in  $\vec{b}_j$ . Otherwise,  $A_d$  rejects  $\vec{b}_j$  and judges that not all packets claimed in  $\vec{b}_j$  are actually received by  $n_j$ , so  $n_j$  is a malicious node. We prove the correctness of this auditing algorithm in Section 4.3.

Note that the above mechanism only guarantees that a node cannot understate its packet loss, i.e., it cannot claim the reception of a packet that it actually did not receive. This mechanism cannot prevent a node from overly stating its packet loss by claiming that it did not receive a packet that it actually received. This latter case is prevented by another mechanism discussed in the detection phase.

#### 4.2.4 Detection Phase

The public auditor  $A_d$  enters the detection phase after receiving and auditing the reply to its challenge from all nodes on  $P_{SD}$ . The main tasks of  $A_d$  in this phase include the following: detecting any overstatement of packet loss at each node, constructing a packet-loss bitmap for each hop, calculating the autocorrelation function for the packet loss on each hop, and deciding whether malicious behavior is present. More specifically,  $A_d$  performs these tasks as follows.

Given the packet-reception bitmap at each node,  $\vec{b}_1, \dots, \vec{b}_K$ ,  $A_d$  first checks the consistency of the bitmaps for any possible overstatement of packet losses. Clearly, if there is no overstatement of packet loss, then the set of packets received at node  $j+1$  should be a subset of the packets received at node  $j$ , for  $j = 1, \dots, K-1$ . Because a normal node always truthfully reports its packet reception, the packet-reception bitmap of a malicious node that overstates its packet loss must contradict with the bitmap of a normal downstream node. Note that there is always at least one normal downstream node, i.e., the destination  $D$ . So  $A_d$  only needs to sequentially scan  $\vec{b}_j$ 's and the report from  $D$  to identify nodes that are overstating their packet losses.

After checking for the consistency of  $\vec{b}_j$ 's,  $A_d$  starts constructing the per-hop packet-loss bitmap  $\vec{m}_j$  from  $\vec{b}_{j-1}$  and  $\vec{b}_j$ . This is done sequentially, starting from the first hop from  $S$ . In each step, only packets that are lost in the current hop will be accounted for in  $m_j$ . The packets that were not received by the upstream node will be marked as

“not lost” for the underlying hop. Denoting the “lost” packet by 0 and “not lost” by 1,  $\vec{m}_j$  can be easily constructed by conducting a bit-wise complement-XOR operation of  $\vec{b}_{j-1}$  and  $\vec{b}_j$ . For example, consider the following simple case with three intermediate nodes (four hops) on the route and with  $M = 10$ . Suppose that  $\vec{b}_1 = (0, 1, 1, 1, 1, 1, 1, 0, 1)$ ,  $\vec{b}_2 = (0, 1, 1, 1, 1, 1, 1, 0, 1)$ ,  $\vec{b}_3 = (0, 1, 0, 1, 1, 0, 1, 1, 0, 1)$ , and the destination  $D$  reports that  $\vec{b}_D = (0, 1, 0, 1, 1, 0, 1, 1, 0, 1)$ . Then the per-hop packet-loss bitmaps are given by  $\vec{m}_1 = (0, 1, 1, 1, 1, 1, 1, 0, 1)$ ,  $\vec{m}_2 = (1, 1, 1, 1, 1, 1, 1, 1, 1)$ ,  $\vec{m}_3 = (1, 1, 0, 1, 1, 0, 1, 1, 1, 1)$ , and  $\vec{m}_4 = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$ .

The auditor calculates the autocorrelation function  $\gamma_j$  for each sequence  $\vec{m}_j = (m_{j1}, \dots, m_{jM})$ ,  $j = 1, \dots, K$ , as follows

$$\gamma_j(i) = \frac{\sum_{k=1}^{M-i} m_{jk} m_{jk+i}}{M-i}, \quad \text{for } i = 0, \dots, M-1; j = 1, \dots, K. \quad (9)$$

The auditor then calculates the relative difference between  $\gamma_j$  and the ACF of the wireless channel  $f_c$  as follows

$$\epsilon_j = \sum_{i=0}^{M-1} \frac{|\gamma_j(i) - f_c(i)|}{f_c(i)}. \quad (10)$$

The relative difference  $\epsilon_j$  is then used as the decision statistic to decide whether or not the packet loss over the  $j$ th hop is caused by malicious drops. In particular, if  $\epsilon_j \geq \epsilon_{th}$ , where  $\epsilon_{th}$  is an error threshold, then  $A_d$  decides that there is malicious packet drop over the hop. In this case, both ends of the hop will be considered as suspects, i.e., either the transmitter did not send out the packet or the receiver chose to ignore the received packet.  $S$  may choose to exclude both nodes from future packet transmissions, or alternatively, apply a more extensive investigation to refine its detection. For example, this can be done by combining the neighbor-overhearing techniques [9] used in the reputation system. By fusing the testimony from the neighbors of these two nodes,  $A_d$  can pin-point the specific node that dropped the packet. Once being detected, the malicious node will be marked and excluded from the route to mitigate its damage.

The above detection process applies to one end-to-end path. The detection for multiple paths can be performed as multiple independent detections, one for each path. Although the optimal error threshold that minimizes the detection error is still an open problem, our simulations show that through trial-and-error, one can easily find a good  $\epsilon_{th}$  that provides a better detection accuracy than the optimal detection scheme that utilizes only the pdf of the number of lost packets.

**Public Verifiability:** After each detection,  $A_d$  is required to publish the information it received from involved nodes, i.e.,  $\vec{b}_j$ ,  $r^{(j)}$ ,  $s^{(j)}$ , for  $j \in P_{SD}$ , so that a node can verify all calculation has been performed correctly. Note that no knowledge of the HLA secret key  $x$  is required in the verification process. At the same time, because  $A_d$  has no knowledge of  $x$ , there is no way for it to forge a valid HLA signature for  $r^{(j)}$ . In other words,  $A_d$  cannot claim a misbehaving node to be a normal one. Furthermore, the privacy-preserving property of the scheme (see Theorem 4 in Section 4.3) ensures that publishing the auditing information will not compromise the confidentiality of the communication.

### 4.3 Security Analysis

We prove that the proposed scheme has the following security properties.

**Theorem 1:** The verification of  $r^{(j)}$  and  $s^{(j)}$ , as specified in (8), is correct, i.e., (8) must hold for a  $(\vec{c}_j, r^{(j)}, s^{(j)})$  tuple that is constructed according to the specification presented in Section 4.2.3.

*Proof:* The correctness of (8) is shown as follows:

$$\begin{aligned}
e(s^{(j)}, g) &= e\left(\prod_{i=1, b_{ji} \neq 0}^M s_{ji}^{c_{ji}}, g\right) \\
&= e\left(\prod_{i=1, b_{ji} \neq 0}^M \{H_2(i||j)u^{r_i}\}^{xc_{ji}}, g\right) \\
&= e\left(\prod_{i=1, b_{ji} \neq 0}^M \{H_2(i||j)u^{r_i}\}^{c_{ji}}, g\right)^x \\
&= e\left(\prod_{i=1, b_{ji} \neq 0}^M H_2(i||j)^{c_{ji}} u^{c_{ji}r_i}, g\right)^x \\
&= e\left(\prod_{i=1, b_{ji} \neq 0}^M H_2(i||j)^{c_{ji}} u^{r^{(j)}}, g^x\right) \\
&= e\left(\prod_{i=1, b_{ji} \neq 0}^M H_2(i||j)^{c_{ji}} u^{r^{(j)}}, v\right). \quad (11)
\end{aligned}$$

So Theorem 1 holds.  $\square$

**Theorem 2:** The construction specified in Section 4.2 is secure under the collusion model defined in Section 3.2, i.e., an adversary that does not receive a packet  $P_i$  cannot claim receiving this packet in its  $\vec{b}_j$  by forging a HLA signature for a random linear combination of the received packets, even if this adversary colludes with any other malicious node in  $P_{SD}$ .

*Proof:* For a given node  $n_j$ , our construction essentially follows the BLS-signature-based HLA construction described in [24]. Under the implicitly assumed condition of no collusion between attackers, the authors in [24] proved that the construction is secure, i.e., no adversary can forge a response to a random challenge if it does not know the HLA signature of each packet in the linear combination. So here, we only need to show that collusion between malicious nodes does not give the attacker more information about the HLA signature of the packets. This can be shown by observing the following novel properties of our HLA construction:

1. For a packet  $P_i$ , the signature scheme specified in (3) dictates that its HLA signature  $s_{ji}$  is not only tied to the packet sequence number ( $i$ ), but also related to the node index ( $j$ ) that is relaying the packet. This means that for the same packet, each hop on  $P_{SD}$  is given a different HLA signature. The verification scheme in (8) accounts for both  $i$  and  $j$ . In the no-collusion case, by treating the concatenation of  $(i||j)$  as a meta packet sequence number, the security of our construction can be proved in the same way as that in [24].
2. The way that the HLA signatures are distributed to nodes on  $P_{SD}$ , as specified in (4), dictates that an upstream node  $n_j$  cannot get a copy of the HLA signature  $s_{j'i}$  of a downstream node  $n_{j'}$ , where  $j < j' \leq K$ , unless the downstream node  $n_{j'}$  receives the signature

$s_{j'i}$  first on  $P_{SD}$  and then sends it through the covert channel to the upstream node  $n_j$ . Therefore, there is no way for a downstream malicious node to get any information on its HLA signature if the upstream attacker drops the packet. As a result, the secret information exchange on the covert channel does not help the adversary to get more information on its HLA signature than the scenario where there is no collusion.

Combining the above arguments, Theorem 2 is proved.  $\square$

**Theorem 3:** The proposed scheme ensures that the packet-reception bitmap reported by a node in  $P_{SD}$  is truthful.

The validity of Theorem 3 is straightforward, because Theorem 2 guarantees that the node cannot understate its packet loss information. At the same time, from our discussion in Section 4.2.4, it is clear that a malicious node cannot overstate its packet loss either. So a node must report its actual packet reception information truthfully to  $A_d$ .

**Theorem 4:** Our HLA construction is publicly verifiable and privacy preserving, i.e., the auditor  $A_d$  does not require the secret key of the HLA scheme to verify a node's response. In addition,  $A_d$  cannot determine the content of the packets transmitted over  $P_{SD}$  from the information submitted by nodes.

*Proof:* Public verifiability is clear from the construction of the scheme. The privacy-preserving property is guaranteed by the application of the secure hash function  $H_1$ . More specifically, instead of directly computing the HLA signature for a packet  $P_i$ , our construction computes the signature for the image of the packet  $r_i = H_1(P_i)$ . During the auditing phase,  $A_d$  can collect a set of linear combinations of  $r_i$ 's. So it is possible for  $A_d$  to calculate  $r_i$ 's by solving a set of linear equations, if a sufficient number of combinations are collected. Even if  $A_d$  can recover  $r_i$ , it should not be able to guess  $P_i$  because of  $H_1$ 's resilience to the pre-image attack.  $\square$

### 4.4 Overhead Analysis

The proposed scheme requires relatively high computation capability at the source, but incurs low communication and storage overheads along the route, as explained below.

#### 4.4.1 Computation Requirements

Most of the computation is done at the source node (for generating HLA signatures) and at the public auditor (for conducting the detection process). We consider the public auditor as a dedicated service provider that is not constrained by its computing capacity. So the computational overhead should not be a factor limiting the application of the algorithm at the public auditor. On the other hand, the proposed algorithm requires the source node to generate  $K$  HLA signatures for a  $K$ -hop path for each data packet. The generation of HLA signatures is computationally expensive, and may limit the applicability of the algorithm. One solution to this problem is to make the signature scalable, e.g., instead of generating a per-packet signature, a per-block signature may be generated, where each block has  $L$  packets. Accordingly, the detection will be extended to blocks (a block is defined as lost if a packet in the block is lost). This could significantly reduce the computational overhead at the source. This method will be evaluated in our future work.

#### 4.4.2 Communication Overhead

The communication overhead for the setup phase is a one-time cost, incurred when  $P_{SD}$  is established. Here we mainly focus on the recurring cost during the packet transmission and auditing phases (there is no communication overhead in the detection phase). For a transmitted packet  $P_i$ ,  $S$  needs to send one encrypted HLA signature and one MAC to each intermediate node on  $P_{SD}$ . Our HLA signature follows the BLS scheme in [5]. So an HLA signature  $s_{ij}$  is 160-bit long. If encrypted by DES, the encrypted signature  $\tilde{s}_{ij}$  is 192 bits in length (a block in DES is 64-bit long, so the length of the cipher text of DES is multiples of 64 bits). The MAC-related hash function  $H_{key}^{MAC}$  can be implemented in SHA-1 and has a length of 160 bits. So for each packet, the per-hop communication overhead incurred by the proposed scheme in the packet transmission phase is  $192 + 160 = 352$  bits, or 44 bytes. For a path of  $K$  intermediate hops, the total communication overhead for transmitting a packet is  $44K$  bytes. For example, when  $K = 10$ , the overhead is 440 bytes/packet. For an IEEE 802.11 system, this is about 19% of the maximum MSDU (2304 bytes).

In the auditing phase, the auditor  $A_d$  sends a random challenge vector  $\vec{c}_j$  to each node  $n_j$ . Let each element in this vector be a 32-bit integer. The challenge has a length of  $4M$  bytes. Based on our simulation in Section 5,  $M = 50$  is typically enough to achieve good detection accuracy. So this means each challenge can be delivered in one packet. Node  $n_j$  replies to the challenge with  $\vec{b}_j$ ,  $r^{(j)}$ , and  $s^{(j)}$ . Among them,  $\vec{b}_j$  is an  $M$ -bit bitmap.  $r^{(j)}$  is the linear combination of the SHA-1 image of the packets, so  $r^{(j)}$  also has a length of 160 bits.  $s^{(j)}$  is an HLA signature of  $r^{(j)}$ , so it is also 160-bit long. Overall, the reply from a node to  $A_d$  has a length of  $320 + M$  bits, which can also be delivered in one packet.

#### 4.4.3 Storage Overhead

During its operation, a node  $n_j$  on  $P_{SD}$  needs to store the key  $key_j$ , the  $H_1$  hash image, and the associated HLA signature for each of the  $M$  most recently received packets. Assuming  $encrypt_{key}$  and  $decrypt_{key}$  are based on DES,  $key_j$  has a length of 56 bits. Let the hash function  $H_1$  be based on SHA-1. So the  $H_1$  image of a packet is 160-bit long. The HLA signature is based on BLS (Boneh-Lynn-Shacham) scheme [5] and is 160-bit long. So in total the storage overhead at  $n_j$  is  $320M + 56$  bits, or  $40M + 7$  bytes. This storage overhead is quite low. For example, when  $M \leq 50$ , the storage overhead at a node is less than 2 KB.

## 5. PERFORMANCE EVALUATION

### 5.1 Simulation Setup

In this section, we compare the detection accuracy achieved by the proposed algorithm with the optimal maximum likelihood (ML) algorithm, which only utilizes the distribution of the number of lost packets. For given packet-loss bitmaps, the detection on different hops is conducted separately. So, we only need to simulate the detection of one hop to evaluate the performance of a given algorithm. We assume packets are transmitted continuously over this hop, i.e., a saturated traffic environment. We assume channel fluctuations for this hop follow the Gilbert-Elliot model, with the transition probabilities from good to bad and from bad to good given by  $P_{GB}$  and  $P_{BG}$ , respectively. We consider two types of malicious packet dropping: random dropping and selec-

tive dropping. In the random dropping attack, a packet is dropped at the malicious node with probability  $P_M$ . In the selective dropping attack, the adversary drops packets of certain sequence numbers. In our simulations, this is done by dropping the middle  $N$  of the  $M$  most recently received packets, i.e., setting the  $N$  bits in the middle of the packet-loss bitmap to 0 (if a packet in these positions is dropped due to link errors, then the set of 0's extends to an extra bit in the middle).  $P_M$  and  $N$  are simulation parameters that describe the selectivity of the attack. In both cases, we let  $\epsilon_{th} = 10\%$  for the proposed algorithm.

We are interested in the following three performance metrics: probability of false alarm ( $P_{fa}$ ), probability of miss-detection ( $P_{md}$ ), and the overall detection-error probability ( $P_{error}$ ). We collect these statistics as follows. In each run, we first simulate 1000 independently generated packet-loss bitmaps for the hop, where packet losses are caused by link errors only. We execute our detection algorithm over these packet-loss bitmaps and collect the number of cases where the algorithm decides that an attacker is present. Let this number be  $I_{fa}$ .  $P_{fa}$  of this run is calculated as  $P_{fa} = I_{fa}/1000$ . We then simulate another 1000 independently generated packet-loss bitmaps, where losses are now caused by both link errors and malicious drops. Let the number of cases where the detection algorithm rules that an attacker is not present be  $I_{md}$ .  $P_{md}$  of the underlying run is given by  $P_{md} = I_{md}/1000$ .  $P_{error}$  is given by  $P_{error} = (I_{fa} + I_{md})/2000$ . The above simulation is repeated 30 times, and the mean and 95% confidence interval are computed for the various performance metrics.

## 5.2 Results

### 5.2.1 Random Packet Dropping

The detection accuracy is shown in Figure 3 as a function of the malicious random-drop rate  $P_M$ . In each subfigure, there are two sets of curves, representing the proposed algorithm and the optimal ML scheme, respectively. In each set of curves, the one in the middle represents the mean, and the other two represent the 95% confidence interval. In general, the detection accuracy of both algorithms improves with  $P_M$  (i.e., the detection error decreases with  $P_M$ ). This is not surprising, because malicious packet drops become more statistically distinguishable as the attacker starts to drop more packets. In addition, this figure shows that for  $\epsilon_{th} = 10\%$ , the proposed algorithm provides slightly higher false-alarm rate (subfigure (c)) but significantly lower miss-detection probability (subfigure (b)) than the ML scheme. A low miss-detection probability is very desirable in our context, because it means a malicious node can be detected with a higher probability. The slightly higher false-alarm rate should not be a problem, because a false alarm can be easily recognized and fixed in the post-detection investigation phase. Most importantly, the overall detection-error probability of the proposed scheme is lower than that of the ML scheme (subfigure (a)). We are especially interested in the regime when  $P_M$  is comparable to the average packet loss rate due to link errors, given by  $\frac{P_{GB}}{P_{GB}+P_{BG}} = \frac{0.01}{0.01+0.5} \approx 0.02$ . This regime represents the scenario in which the attacker hides its drops in the background of link errors by mimicking the channel-related loss rate. In this case, the ML scheme cannot correctly differentiate between link errors and malicious drops. For example, when  $P_M = 0.01$ , the ML scheme results in  $P_{md} = 80\%$  and  $P_{fa} = 23\%$ . This is close to arbitrarily ruling that every packet loss is due to link error



only, leading to an overall detection-error rate of 50% (see subfigure (a)). Our proposed algorithm, on the other hand, achieves a much better detection accuracy, because its  $P_{md}$  and  $P_{fa}$  are both lower than those under the ML scheme. As a result, when  $P_M = 0.01$ , the total detection-error rate of the proposed algorithm is about 35%. When  $P_M$  is increased to 0.04,  $P_{error}$  of the proposed scheme reduces to only 20%, which is roughly half of the error rate of the ML scheme at the same  $P_M$ . Remembering that the detection-error rate of the ML scheme is the lowest among all detection schemes that only utilize the distribution of the number of lost packets, the lower detection-error rate of the proposed scheme shows that exploiting the correlation between lost packets helps in identifying the real cause of packet drops more accurately. The effect of exploiting the correlation is especially visible when the malicious packet-drop rate is comparable with the link error rate.

In Figure 4, we plot the detection accuracy as a function of the size of the packet-loss bitmap ( $M$ ). It can be observed that  $P_{error}$  for the proposed scheme decreases with  $M$ . However, as  $M$  becomes sufficiently large, e.g.,  $M = 30$  in our case, a further increase in the size of the bitmap does not lead to additional improvement in the detection accuracy. This can be explained by noting that the two-state Markovian GE channel model has a short-range dependence, i.e., the correlation between two points of the fluctuation process decays rapidly with the increase in the separation between these points. This short-range dependence is reflected in an exponentially decaying autocorrelation function for the channel. As a result, a good estimation of the autocorrelation function can be derived as long as  $M$  is long enough to cover the function's short tail. This phenomenon implies that a node does not need to maintain a large packet-reception database in order to achieve a good detection accuracy under the proposed scheme. It also explains the low storage overhead incurred by our scheme.

The detection accuracy is plotted in Figure 5 as a function of the channel state transition rate  $P_{GB}$ . It can be observed from this figure that  $P_{error}$  for both algorithms increases with  $P_{GB}$ . This is not surprising because at its initial point of  $P_{GB} = 0.01$ , the expected link error rate is about 0.02, which is much smaller than the malicious packet drop rate of  $P_M = 0.1$ . So it is relatively easy to differentiate between the case where packet drops are caused by link errors only and the one where such drops are caused by the combined effect of link errors and malicious drops. As  $P_{GB}$  increases, the link error probability approaches  $P_M$ , making the statistical separation of the two cases harder. As a result, the detection error increases with  $P_{GB}$ . For all values of  $P_{GB}$  in this figure, the proposed algorithm always achieves significantly lower detection-error probability than the ML scheme.

### 5.2.2 Selective Packet Dropping

The detection error as a function of the number of maliciously dropped packets is shown in Figure 6. At the low end of the  $x$ -axis, maliciously dropped packets account for only  $1/50 = 2\%$  of the total packets in the packet-loss bitmap. This is identical to the link error rate of 0.02, assumed in the simulation. Similar performance trends can be observed to the case of the random packet dropping. Fewer detection errors are made by both algorithms when more packets are maliciously dropped. In all the simulated cases, the proposed algorithm can detect the actual cause of the packet drop more accurately than the ML scheme, especially when the number of maliciously dropped packets is small. When

the number of maliciously dropped packets is significantly higher than that caused by link errors (greater than 4 packets in our simulation), the two algorithms achieve comparable detection accuracy. In this scenario, it may be wise to use the conventional ML scheme due to its simplicity (e.g., no need to enforce truthful reports from intermediate nodes, etc).

The detection errors are plotted in Figure 7 as a function of the size of the packet-loss bitmap ( $M$ ). To conduct a fair comparison, as we increase  $M$ , we also increase the number of maliciously dropped packets, so as to maintain a malicious packet-dropping rate of 10%. It can be observed that a small  $M$  is enough to achieve good detection accuracy under the proposed scheme, due to the short-range dependence property of the channel.

In Figure 8, the detection errors are plotted as a function of the channel state transition probability  $P_{GB}$ . Similar trends are observed to those in the random packet dropping case, i.e., the algorithms make more detection errors when the link error rate approaches the malicious packet-drop rate. Once again, the proposed algorithm consistently outperforms the ML scheme in all the tested cases.

## 6. CONCLUSIONS

In this paper, we showed that compared with conventional detection algorithms that utilize only the distribution of the number of lost packets, exploiting the correlation between lost packets significantly improves the accuracy in detecting malicious packet drops. Such improvement is especially visible when the number of maliciously dropped packets is comparable with those caused by link errors. To correctly calculate the correlation between lost packets, it is critical to acquire truthful packet-loss information at individual nodes. We developed an HLA-based public auditing architecture that ensures truthful packet-loss reporting by individual nodes. This architecture is collusion proof, requires relatively high computational capacity at the source node, but incurs low communication and storage overheads over the route.

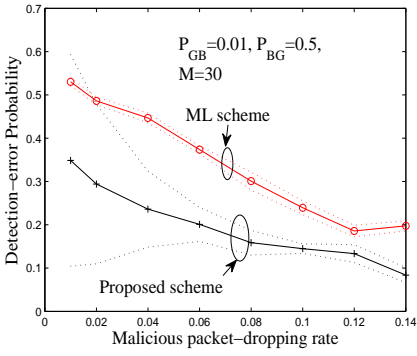
Some open issues remain to be explored. First, the computational overhead at source nodes needs to be reduced. As we pointed out in Section 4.4.1, a block-based HLA signature could be explored. We will evaluate the effect of this method as our next step. Second, in this paper, we mainly focused on showing the feasibility of the proposed mechanism. The decision threshold used in the detection was obtained by trial-and-error. In our future work, we will study the optimization of this threshold. Last but not least, the proposed detection algorithm does not account for topological changes in the network. The impact of dynamic topology remains an issue to be evaluated.

## 7. ACKNOWLEDGMENTS

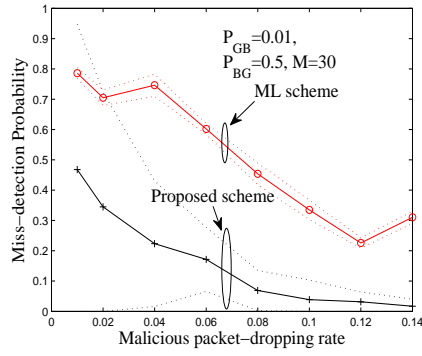
This research was supported in part by NSF (under grants CNS-1016943, CNS-0904681, and IIP-0832238), Raytheon, and the ařConnection Oneař center. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## 8. REFERENCES

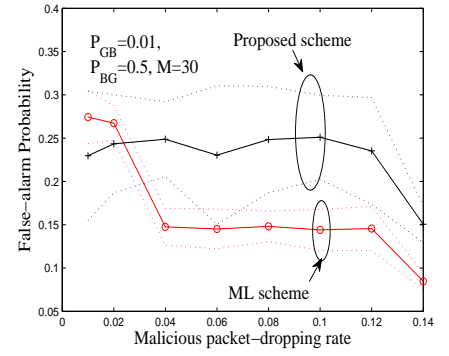
- [1] J. N. Arauz. 802.11 Markov channel modeling. *Ph.D. Dissertation, School of Information Science, University of Pittsburgh*, 2004.



(a) Overall detection-error probability

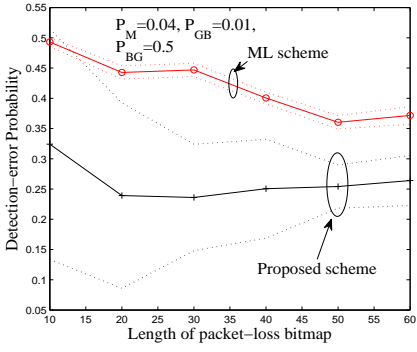


(b) Miss-detection probability

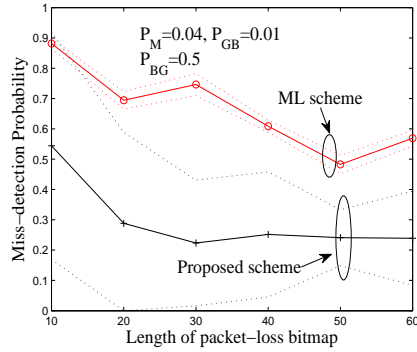


(c) False-alarm probability

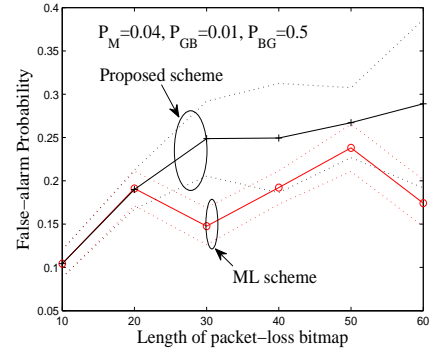
Figure 3: Detection accuracy vs.  $P_M$  (random packet-drop case).



(a) Overall detection-error probability

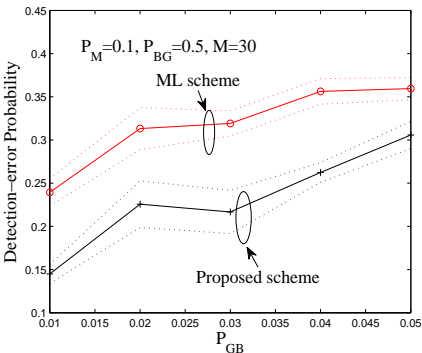


(b) Miss-detection probability

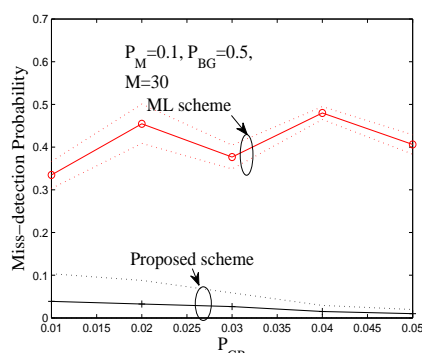


(c) False-alarm probability

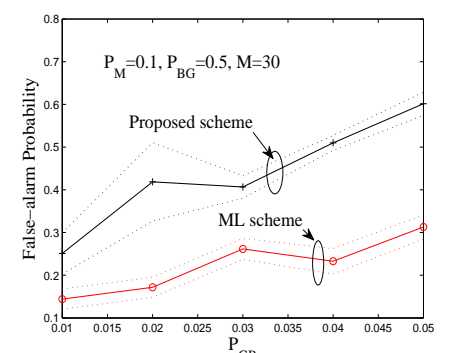
Figure 4: Detection accuracy vs.  $M$  (random packet-drop case).



(a) Overall detection-error probability

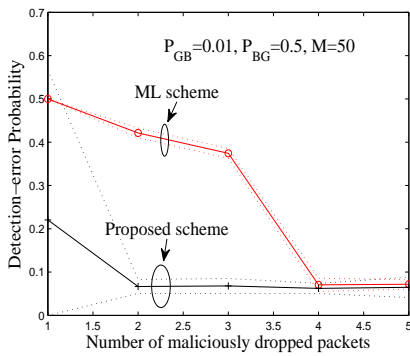


(b) Miss-detection probability

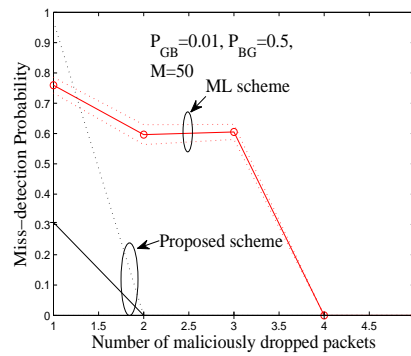


(c) False-alarm probability

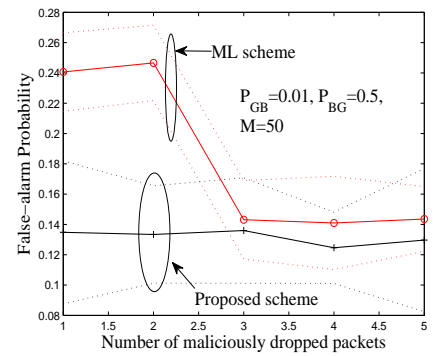
Figure 5: Detection accuracy vs.  $P_{GB}$  (random packet-drop case).



(a) Overall detection-error probability

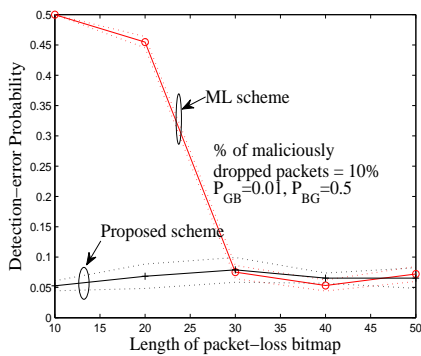


(b) Miss-detection probability

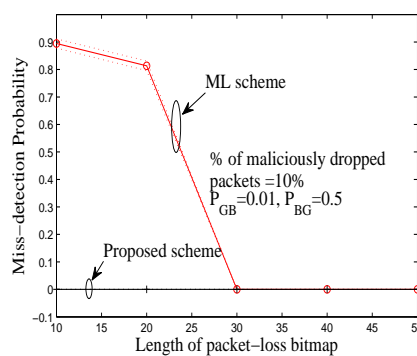


(c) False-alarm probability

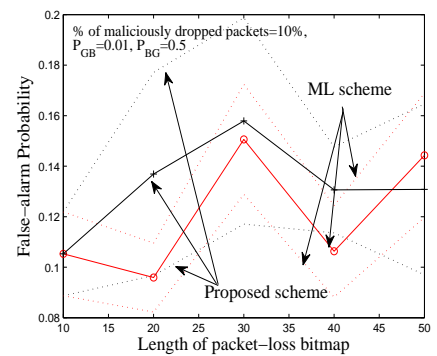
**Figure 6: Detection accuracy vs. number of maliciously dropped packets (selective packet-drop case).**



(a) Overall detection-error probability

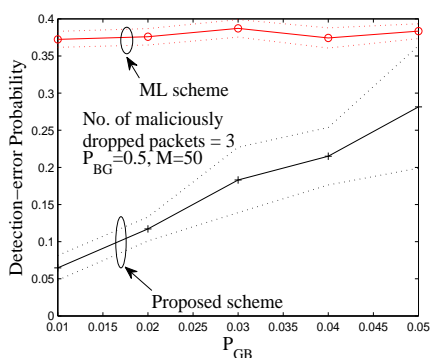


(b) Miss-detection probability

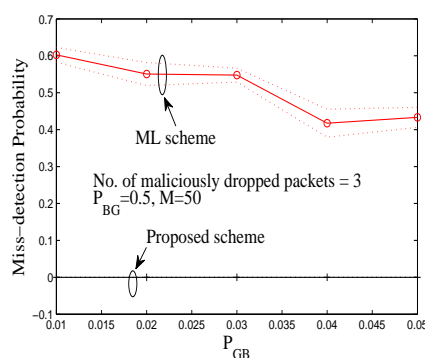


(c) False-alarm probability

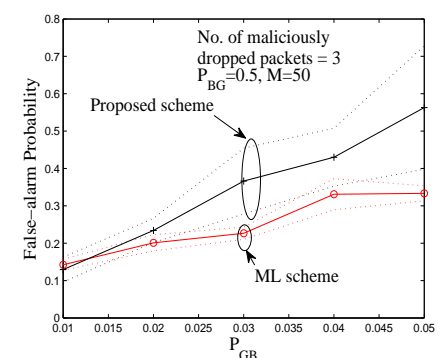
**Figure 7: Detection accuracy vs.  $M$  (selective packet-drop case).**



(a) Overall detection-error probability



(b) Miss-detection probability



(c) False-alarm probability

**Figure 8: Detection accuracy vs.  $P_{GB}$  (selective packet-drop case).**

- [2] C. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 598–610, Oct. 2007.
- [3] G. Ateniese, S. Kamara, and J. Katz. Proofs of storage from homomorphic identification protocols. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, 2009.
- [4] B. Awerbuch, R. Curtmola, D. Holmer, C. Nita-Rotaru, and H. Rubens. ODSBR: an on-demand secure byzantine resilient routing protocol for wireless ad hoc networks. *ACM TISSEC*, 10(4), 2008.
- [5] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. *Journal of Cryptology*, 17(4):297–319, Sept. 2004.
- [6] S. Buchegger and J. Y. L. Boudec. Performance analysis of the confidant protocol (cooperation of nodes: fairness in dynamic ad-hoc networks). In *Proceedings of the ACM MobiHoc Conference*, 2002.
- [7] L. Buttyan and J. P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *ACM/Kluwer Mobile Networks and Applications*, 8(5):579–592, Oct. 2003.
- [8] J. Eriksson, M. Faloutsos, and S. Krishnamurthy. Routing amid colluding attackers. 2007.
- [9] W. Galuba, P. Papadimitratos, M. Poturalski, K. Aberer, Z. Despotovic, and W. Kellerer. Castor: Scalable secure routing for ad hoc networks. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, march 2010.
- [10] T. Hayajneh, P. Krishnamurthy, D. Tipper, and T. Kim. Detecting malicious packet dropping in the presence of collisions and channel errors in wireless ad hoc networks. In *Proceedings of the IEEE ICC Conference*, 2009.
- [11] Q. He, D. Wu, and P. Khosla. Sori: a secure and objective reputation-based incentive scheme for ad hoc networks. In *Proceedings of the IEEE WCNC Conference*, 2004.
- [12] D. B. Johnson, D. A. Maltz, and J. Broch. DSR: the dynamic source routing protocol for multi-hop wireless ad hoc networks. *Chapter 5, Ad Hoc Networking, Addison-Wesley*, pages 139–172, 2001.
- [13] W. Kozma Jr. and L. Lazos. Dealing with liars: misbehavior identification via Renyi-Ulam games. In *Proceedings of the International ICST Conference on Security and Privacy in Communication Networks (SecureComm)*, 2009.
- [14] W. Kozma Jr. and L. Lazos. REAct: resource-efficient accountability for node misbehavior in ad hoc networks based on random audits. In *Proceedings of the ACM Conference on Wireless Network Security (WiSec)*, 2009.
- [15] K. Liu, J. Deng, P. Varshney, and K. Balakrishnan. An acknowledgement-based approach for the detection of routing misbehavior in MANETs. *IEEE Transactions on Mobile Computing*, 6(5):536–550, May 2006.
- [16] Y. Liu and Y. R. Yang. Reputation propagation and agreement in mobile ad-hoc networks. In *Proceedings of the IEEE WCNC Conference*, pages 1510–1515, 2003.
- [17] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the ACM MobiCom Conference*, pages 255–265, 2000.
- [18] G. Noubir and G. Lin. Low-power DoS attacks in data wireless lans and countermeasures. *ACM SIGMOBILE Mobile Computing and Communications Review*, 7(3):29–30, July 2003.
- [19] V. N. Padmanabhan and D. R. Simon. Secure traceroute to detect faulty or malicious routing. In *Proceedings of the ACM SIGCOMM Conference*, 2003.
- [20] P. Papadimitratos and Z. Haas. Secure message transmission in mobile ad hoc networks. *Ad Hoc Networks*, 1(1):193–209, 2003.
- [21] A. Proano and L. Lazos. Selective jamming attacks in wireless networks. In *Proceedings of the IEEE ICC Conference*, pages 1–6, 2010.
- [22] A. Proano and L. Lazos. Packet-hiding methods for preventing selective jamming attacks. *IEEE Transactions on Dependable and Secure Computing*, 9(1):101–114, 2012.
- [23] R. Rao and G. Kesidis. Detecting malicious packet dropping using statistically regular traffic patterns in multihop wireless networks that are not bandwidth limited. In *Proceedings of the IEEE GLOBECOM Conference*, 2003.
- [24] H. Shacham and B. Waters. Compact proofs of retrievability. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT)*, Dec. 2008.
- [25] C. Wang, Q. Wang, K. Ren, and W. Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *Proceedings of the IEEE INFOCOM Conference*, Mar. 2010.
- [26] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *Proceedings of the ACM MobiHoc Conference*, pages 46–57, 2005.
- [27] S. Zhong, J. Chen, and Y. R. Yang. Sprite: a simple cheat-proof, credit-based system for mobile ad-hoc networks. In *Proceedings of the IEEE INFOCOM Conference*, pages 1987–1997, 2003.