Witness-based Detection of Forwarding Misbehaviors in Wireless Networks

UMass Computer Science Technical Report UM-CS-2009-001

Sookhyun Yang*, Sudarshan Vasudevan*, and Jim Kurose*

*Dept. Computer Science, University of Massachusetts, Amherst, MA, 01003, Email: {shyang, svasu, kurose}@cs.umass.edu

***Abstract*—We consider the problem of identifying a node that incorrectly forwards packets in a static wireless ad hoc network. We propose a detection scheme that identifies a misbehaving node based on observations made by neighboring nodes ("witnesses") near the forwarding node. Without longterm or cumulative observation, the proposed scheme makes an instantaneous decision about whether a node is correctly forwarding a packet. Through extensive analysis of our scheme under various threat scenarios, we show that our scheme can unambiguously identify a misbehaving node when there is no collusion and can detect the existence of misbehaving nodes in the presence of collusion. We analyze our scheme's detection accuracy and communication overhead accounting for the lossy nature of wireless links. Our analysis demonstrates that our scheme can achieve high detection accuracy while incurring low communication overhead.**

I. Introduction

In a wireless ad hoc network, each node transmits data packets to a destination via intermediate nodes on a path. A number of secure ad hoc routing protocols [6] have been proposed for guaranteeing secure end-end data transmission. In these solutions, signed routing messages prevent an unauthenticated node from joining a network. However, these schemes suffer from one important drawback: if a previously authenticated node becomes compromised, there are no safeguards in place to identify and thwart adversaries that misuse the compromised node. Consequently, a path verification mechanism (i.e., a mechanism to ensure that each node along an end-end path is correctly forwarding packets) is necessary to observe and identify such a misbehaving node. The detection of compromised nodes is particularly important in military MANETs [11, 19, 20] and other networks in which nodes can be compromised either physically or remotely.

Path verification mechanisms can be classified into two categories [9, 14]: control-plane verification mechanisms, and data-plane verification mechanisms. Control-plane verification mechanisms detect routing disruption attacks that inject false routing control messages. Data-plane verification deals with data forwarding misbehavior attacks such as packet drops, reordering, and message corruption. In this paper, we focus on solutions to detect forwarding misbehavior in the data plane. As we will see, in addition to detecting the presence of an attack, our solution also identifies the source of forwarding misbehavior.

The main contributions of this paper can be summarized as follows:

- We present a witness-based detection scheme that uses multiple observing nodes to make a precise and immediate detection of a compromised node that incorrectly forwards packets.
- We show that our scheme can unambiguously identify a compromised node, as long as there is at least one uncompromised observing node and compromised nodes do not collude.
- For the case when compromised nodes do collude, we show that our scheme can detect the existence of compromised nodes, as long as there is at least one uncompromised observing node.
- Using an analytical model, we show that our scheme can achieve very low false positive and false negative rates in a lossy wireless network.
- Furthermore, our scheme requires relatively low communication overhead while achieving high accuracy.

The rest of this paper is organized as follows. In section II, we define our data-path verification model and forwarding misbehavior, and describe two detection schemes. In section III, we describe the witness-based detection scheme in more detail. Section IV shows how our scheme detects attacks in various threat scenarios. In section V, we evaluate our scheme's detection accuracy and communication overhead in the presence of lossy links. In section VI, we discuss related work. In the last section, we conclude our paper.

II. Detecting Forwarding Misbehavior

Before we discuss witness-based detection in detail, we define various forwarding misbehavior attacks and introduce a basic data-path verification model for detecting forwarding misbehavior. We briefly describe two detection schemes: data-path-based detection and witness-based detection.

A. Forwarding misbehavior attacks

We consider a static wireless ad hoc network that is composed of authenticated nodes using public-key authentication and assume that each node in a network has a valid public/private key pair and the public keys for all nodes are publicly known. We consider the case that an authenticated node is attacked (compromised) and its private key is stolen.

The compromised node then launches one or more of the following forwarding misbehavior attacks:

- *Drop (Blackhole or Grayhole)* is an attack in which a compromised node does not forward a data packet. This drop attack includes complete, partial, or selective dropping of packets.
- *Fake forwarding* is an attack in which a compromised node forwards a data packet to a nonexistent node.
- *Route deviation* is an attack in which a compromised node forwards a data packet to an incorrect next-hop neighbor. This attack results in a data packet failing to reach its destination or causes delay by forwarding a data packet off the optimal data path.
- *Power control* is an attack in which a compromised node forwards a data packet with insufficient transmission power, causing the data packet to be unreachable to its nexthop neighbor on the data path.
- *Reorder (Jellyfish)* is an attack in which a compromised node forwards a data packet out-of-order.
- *Message corruption* is an attack in which a compromised node corrupts a message field in a data packet.

B. Data-path verification model

We introduce a data-path forwarding verification model that mimics a real-world court trial. In this model, each node on a data path verifies whether its direct downstream neighboring node (with respect to some destination) is correctly forwarding data packets. The upstream node detects its downstream neighboring node's state as compromised or uncompromised, depending on the correctness of forwarding behavior. Without depending on longterm cumulative observation as in [2, 5, 12, 13, 15, 17, 21], our approach provides an instantaneous detection on a packet-by-packet basis. The following five components constitute the model.

- *Defendant* is an intermediate node on a data path that is supposed to forward data packets.
- *Defendant's downstream neighbor* is a defendant's direct downstream neighbor on a data path. This node should receive a data packet from the defendant.
- *Evidence* is an observation of a defendant's forwarding behavior. Different evidence exists for each packet.
- *Judge* is a defendant's *upstream* node that decides the defendant's state (compromised, uncompromised) based on received evidence. If a judge does not receive valid evidence within a defined time, the judge determines the defendant's state as compromised.
- *Witness* is a node located near a defendant (except the defendant's downstream neighbor) that can overhear message transmissions that provide evidence of the defendant's forwarding behavior. By transmitting evidence to a judge, a witness helps a judge make its decision.

In contrast to [2, 5, 12, 13, 15, 17, 21] *where a witness decides a defendant's state, our model distinguishes a judge from a witness and places the decision regarding the defendant node's state (compromised, uncompromised) in the hands of the judge rather than the witness*. Unlike witness nodes, an upstream node has information regarding the correct forwarding behavior such as the route to a destination and the sequence of data packets. Thus, the upstream node can provide better detection accuracy than a witness node alone.

This model can be implemented in a centralized or decentralized manner. We focus on the decentralized case in this paper.

C. Two forwarding misbehavior detection schemes

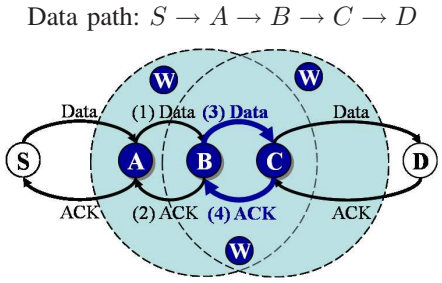


Fig. 1: **Reliable hop-by-hop data forwarding ($Data/ACK$ exchange) in a wireless ad hoc network**

Considering the data-path verification model described above, we now describe two forwarding misbehavior detection schemes for reliable hop-by-hop data forwarding, where each node on a data path exchanges $Data$ and ACK packets with its next-hop neighbor as part of the normal forwarding of packets. Without loss of generality, we consider a data path $S \rightarrow A \rightarrow B \rightarrow C \rightarrow D$, where S is a source, D the ultimate destination, and A, B, C are intermediate nodes, as shown in Fig. 1.

From now on, we focus on a defendant node B's verification by its upstream node A as a judge. In Fig. 1, each number in parentheses depicts the ordering of $Data$ and ACK packet exchange. We use packet (3) and packet (4) as evidence of defendant B's forwarding (Here, we do not consider packet (2) as evidence for forwarding behavior, since node B's replying ACK to node A is irrelevant to node B's actual downstream forwarding behavior). A witness is a node who may overhear packet (3) or (4), and is located within node B or node C's transmission radius depicted by dotted lines in Fig. 1. Let W be the set of witness nodes excluding node B and node C.

We have two forwarding misbehavior detection schemes, which differ in the role of witness nodes.

- **Data-path-based detection** was suggested in [1, 7, 9, 10, 16]. Without intervention of witnesses, data-path-based detection only relies on nodes on the data path. As evidence, the ACK message (i.e., packet (4)) from downstream neighbor C reaches judge node A through defendant node B. As we will see, this ACK will need to be signed by node C. Node A makes a decision on the state of defendant node B using this ACK evidence.
- **Witness-based detection** is a newly proposed scheme in this paper. Witnesses operating in promiscuous mode,

can overhear $Data$ (i.e., packet (3)) as well as ACK (i.e., packet (4)) as evidence and transmit this evidence through diverse paths to judge node A. For decision making, node A utilizes $Data$ and ACK evidence received from both defendant node B and a witness node in set W.

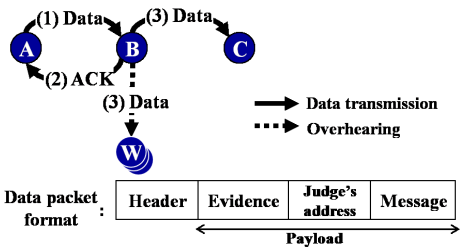
III. Witness-based Detection

In this section, we describe our witness-based detection scheme in detail. We focus our attention on the data path A-B-C as shown in Figure 1. Our witness-based detection scheme consists of three sequential steps as follows:

1) **Evidence generation**: In this step, each node that successfully receives or overhears defendant node B's data packet generates *tamper-proof* evidence of B's forwarding behavior.
2) **Evidence dissemination (ED)**: In this step, evidence-generating nodes reliably transmit evidence to judge node A with low communication overhead.
3) **Judge's decision**: In this step, given the evidence from the ED step, judge node A determines defendant node B's state, or exposes the existence of attacks on the data path or near the data path based on two decision-making algorithms, respectively, as discussed in detail in section III-C below.

We now discuss each of the three steps in detail.

A. Evidence generation



$$B \rightarrow \{C,\ W\}: \quad K_B(r_B),\ \text{addr(A)},\ M \qquad (3)$$

Fig. 2: **Message passing in the evidence-generation step**

Fig. 2 shows the exchange of $Data$ and ACK messages associated with the evidence-generation step. A traditional link-level packet, containing a header and message (payload) field is augmented to carry two additional fields: an evidence field and a judge address field. The judge address field is the address to which evidence (discussed below) regarding this packet is to be sent. The evidence field contains subfields that will ensure that non-forgeable evidence of node B's forwarding behavior can be created only if defendant node B forwards that $Data$ packet. The two subfields of the evidence field are:

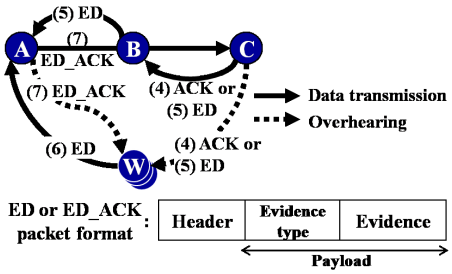
- **Message Checksum**: The checksum is the result of a one-way hash (e.g., MD5) of the message and the message's next-hop recipient (e.g., C) address. We will discuss the use of this checksum field below. The checksum value for a message sent by node B is denoted r_B. This checksum field is further signed by the $Data$ packet forwarder. The signed checksum subfield in a packet sent by node B is denoted $K_B(r_B)$, where K_B is node B's private key.
- **Timestamp**: The time field denotes the time at which the message containing this evidence field was generated and sent, where we assume a global clock in a network, or Lamport-like time ordering. t denotes the timestamp value.

The checksum and timestamp subfields, unique for each packet, are used to prevent replay attacks.

Having described the message format, we next describe the two types of evidence (Data-based and ACK-based). As we will see, Data-based evidence is used to detect every forwarding misbehavior described in Section II.A except the power-control attack, which is detected using ACK-based evidence.

- **Data-based evidence generation**: A witness node w in set W that successfully overhears a $Data$ packet constructs Data-based evidence in the following manner:
  1) extracts $K_B(r_B)$ from node B's *Data* packet.
  2) computes a message checksum (r_w) from node B's $Data$ packet, as described earlier.
  3) records the timestamp (t_w) when it successfully overhears the $Data$ packet.
  4) concatenates and signs these three pieces of information using its private key: $K_w(K_B(r_B), r_w, t_w)$.
- **ACK-based evidence generation**: Using a similar procedure as that employed in Data-based evidence generation, node C generates ACK-based evidence, $K_C(K_B(r_B), r_C, t_C)$.

B. Evidence dissemination



$$C \rightarrow \{B,\ W\} \rightarrow A \quad : 1,\ K_C(K_B(r_B), r_C, t_C) \qquad (5)$$
$$A \rightarrow B \rightarrow C \quad : 1,\ K_A(r_A) \qquad (7)$$

Fig. 3: **Message passing in the evidence-dissemination step**: Note that there is no strict ordering between packet (5) and (6) in this step

In this step, witness nodes and node C that generate evidence transmit Data-based evidence and ACK-based evidence to judge node A, as shown in Fig. 3. Node C transmits ACK-based evidence to node A via defendant node B. Witness nodes use a suppression mechanism (described later) to allow one or a small number of witness nodes to transmit Data-based evidence and relay ACK-based evidence via diverse paths.

For evidence dissemination, we use new control packets, ED and ED_ACK.

- ED **packet**: Node C and each of the witness nodes use an ED packet (i.e., packet (5) and (6)) to transmit evidence. The ED packet contains an evidence type field to distinguish the type of evidence in an evidence field as shown in Fig 3.
- ED_ACK **packet**: Judge node A acknowledges the ED packet using an ED_ACK packet (i.e., packet (7)) that also consists of an evidence type and an evidence field. The ED_ACK packet traverses the data path (A-B-C) to reach all nodes participating in evidence dissemination. Using the evidence type field, node A marks successfully received evidence, which encourages ED_ACK receivers to retransmit evidence that node A does not have. The evidence field contains the message checksum signed by node A (denoted $K_A(r_A)$) and prevents malicious ED_ACK packets being sent by a node other than node A.

As shown in Fig 3, node C's ACK-based evidence to node B can be piggy-backed on an ACK packet (i.e., packet (4)) instead of requiring a separate ED packet.

In a witness node's evidence dissemination, communication overhead during the evidence dissemination step is directly proportional to witness node density. In order to reduce communication overhead, witness nodes perform *randomized feedback suppression*. Witness nodes stay idle for a random time up to a maximum backoff duration. Witness nodes are suppressed if they overhear an ED_ACK packet that contains more than or equal to the amount of evidence that they have to transmit. Highest priority is given to a witness node having both Data-based and ACK-based evidence by setting the shortest maximum backoff duration.

We also consider judge node A's direct overhearing as an auxiliary evidence dissemination method. In this method, node A may directly overhear $Data$ or ACK packets and verify defendant node B's forwarding behavior without additional cost. After successful overhearing of $Data$ or ACK packets, node A transmits an ED_ACK packet.

C. Judge's decision

Before describing judge node A's decision making, we first define variables used in the algorithm in Table I and introduce the notion of *evidence consistency*. Using the variables in the Table I, when node A receives evidence (e_n) associated with $Data$ packet n, node A evaluates $S(e_n)$ using equation (1). Equation (1) first compares the equality of message checksum in the first and second terms. The remaining terms check the correctness of message order. If $S(e_n)$ evaluates to true, we say that the evidence (e_n) is *consistent*. Otherwise, we call e_n *inconsistent*.

$$\begin{aligned} S(e_n) \quad \leftarrow \quad & (r_n = e_n.r_B) \wedge (r_n = e_n.r_x) \\ & \wedge (t_n < e_n.t_x) \wedge (e_{n-1}.t_X < e_n.t_x), \\ & \text{where } e_{n-1}.t_X = \max_{\forall x \in X, 1 \le i \le n-1} e_i.t_x \end{aligned} \quad (1)$$

(a) Variables maintained by node A	
n	Sequence number of $Data$ packet sent by node A
T	Maximum duration for which node A awaits evidence
$addr_n$	Address of a defendant node's nexthop
r_n	Message checksum of $Data$ packet n
t_n	Time when node A sends $Data$ packet n

(b) Evidence variables maintained by node A	
e_n	Received evidence for $Data$ packet n, $e_n = K_x(K_B(r_B), r_x, t_x)$ where $x \in X$ and $X = W \cup \{C\}$
$e_n.addr(x)$	Address of a node x that signed the received evidence
$e_n.r_B$	Message checksum r_B decrypted from B's statement
$e_n.r_x$	Message checksum r_x decrypted from the received evidence
$e_n.t_x$	Timestamp t_x decrypted from the received evidence
$S(e_n)$	Boolean variable set to 1 if e_n is *consistent* and 0 otherwise

TABLE I: Variables used in judge A's decision making algorithm

Node A has two decision-making algorithms based on evidence consistency, depending on whether or not collusion is possible. Algorithm 1 assumes there is no collusion among misbehaving nodes, and determines if node B is compromised, uncompromised or suspicious. Under the assumption of collusion among misbehaving nodes, Algorithm 2 exposes the presence of one or more compromised nodes among node B, node C, and a set W.

The two algorithms make a precise decision using three cheat-proof lemmas below with the following assumptions:

1) Node B has at least one downstream neighbor that can generate ACK-based evidence.
2) Generated evidence successfully reaches node A using reliable paths.
3) At least one uncompromised node $x[\in X]$ (Set X is defined in Table I) is present.

Lemma 1 (Absence of evidence)**.** Absence of evidence implies that defendant node B is compromised.

Proof: The absence of evidence results from defendant B's drop or power control attack, loss of evidence in transit to judge node A, or the failure of a downstream neighbor to reply with an ACK packet. Given assumptions 1) and 2) above, we infer that absence of evidence implies that node B launches a drop or a power control attack. ■

Lemma 2 (Existence of consistency)**.** Suppose that there is no collusion (defined in Section IV). Consistent evidence exists if and only if defendant node B is not compromised.

Proof: ($\Rightarrow$) We first prove that the existence of consistent evidence implies that node B is not compromised by proving its contrapositive version: if node B is compromised, then no consistent evidence can be generated. Suppose that node B is compromised. Without collusion, node x does not know if the sequence of message, message field, or next-hop recipient address in node B's $Data$ packet is correct or not. Node x can generate a message checksum and timestamp either randomly or using the incorrect message field or recipient address of an incorrectly forwarded $Data$ packet by node B. In either case,

$S(e_n)$ evaluates to false by node x's message checksum (r_x) or timestamp (t_x).

($\Leftarrow$) Given assumptions 2) and 3) above, if node B is not compromised, $S(e_n)$ evaluates to true because uncompromised node x puts the correct message checksum and timestamps into evidence, which equal node A's maintained variables in the Table Ia.

Thus, the lemma holds. ■

Lemma 3 (Existence of inconsistency)**.** Existence of inconsistent evidence implies the presence of one or more compromised nodes among nodes B, C, and set W of witness nodes.

Proof: We prove this lemma by using its contrapositive version: if there are no compromised nodes among nodes B, C, and set W, then inconsistent evidence does not exist. Suppose that there are no compromised nodes among nodes B, C and set W. For every evidence generated by nodes B, C, and set W, $S(e_n)$ evaluates to true. Thus, the lemma holds. ■

Algorithm 1 Identify defendant node's state (w/o collusion)

```
1:  S_n ← false
2:  while T > 0 do
3:      S_n ← S_n ∨ S(e_n)
4:      if (S_n = true) ∧ (addr_n = e_n.addr(x)) then
5:          return uncompromised
6:      end if
7:  end while
8:  if S_n = true then
9:      return suspicious
10: end if
11: return compromised
```

In Algorithm 1, if S_n evaluates to true (in other words, there exists at least one consistent piece of evidence) and if the evidence is ACK-based evidence, Algorithm 1 decides that the defendant node is not compromised, in accordance with Lemma 2. However, if S_n is true but every evidence is Data-based evidence, node A classifies node B's state as suspicious, since the absence of ACK-based evidence from node C leaves open to the possibility of a power-control attack by node B, where node B transmits with just enough power to be overheard by the witness nodes, but not high enough to reach node C. Last, if S_n is not true, node B is deemed compromised, based on Lemmas 1 and 2.

Algorithm 2 Expose the existence of attacks (w/ collusion)

```
1:  S_n ← false
2:  while T > 0 do
3:      S_n ← S_n ∨ S(e_n)
4:      if S(e_n) = false then
5:          return existent
6:      end if
7:  end while
8:  if S_n = true then
9:      return non-existent or suspicious
10: end if
11: return existent
```

Algorithm 2 exposes the presence of a malicious attack potentially launched in collusion by two or more among nodes B, C and set W of witness nodes. If inconsistent evidence is received at line 4 or no evidence exists at line 11, Algorithm 2 declares that an attack exists based on Lemma 3 and 1 respectively. On the other hand, if a judge node only receives consistent evidence, Algorithm 2 declares non-existent or suspicious, depending on whether or not ACK-based evidence is present.

IV. Detection Properties

We show how Algorithm 1 identifies a compromised node launching various forwarding misbehaviors described in Section II. Additionally, we show that Algorithms 1 and 2 are invulnerable to three new attacks defined below, which attempt to circumvent the witness-based detection scheme:

- **Bypassing**: Compromised defendant node B that launches forwarding misbehaviors attempts to circumvent the witness-based detection scheme by including an false message checksum or judge's address in a $Data$ packet.
- **Badmouthing**: Compromised node $x[\in X]$ generates evidence that falsely accuses uncompromised defendant node B.
- **Collusion**: Compromised node $x[\in X]$ and compromised node B generate fake consistent evidence together by including a false message checksum or timestamp to conceal node B's forwarding misbehaviors.

A. Forwarding misbehaviors

We describe how our witness-based detection scheme identifies misbehaving defendant node B based on Algorithm 1, when the remaining nodes behave correctly and do not launch the three attacks described above.

- **Drop** attack results from node B not forwarding a $Data$ packet and causes the absence of both Data-based evidence and ACK-based evidence at node A. From Algorithm 1, S_n evaluates to false because node A has no evidence. Thus, node A decides that node B is compromised.
- **Fake forwarding** and **Route deviation** attacks are detected, based on inconsistent Data-based evidence, which contains different message checksums. Let us consider

the scenario in which node B maliciously forwards its $Data$ packet to a node E (as opposed to node C under correct behavior). When node E is non-existent, we term the attack as *fake forwarding*. Otherwise, the attack is called a *route deviation* attack. For every received Data-based evidence e_n=$K_w(K_B(r_B), r_w, t_w)$ (where $w \in W$), it is easy to see that $S(e_n)$ evaluates to false from equation (1), because $r_n \neq e_n.r_B$ and $r_n \neq e_n.r_w$, where $r_n = H[M|addr(C)]$ and $e_n.r_B = e_n.r_w = H[M|addr(E)]$. From Algorithm 1, node A decides that node B is compromised. A route deviation attack is also detectable based on node E's inconsistent ACK-based evidence. Fake forwarding and Route deviation attacks can be distinguished, based on whether or not node A receives ACK-based evidence.

- **Power control** attack allows for node A to receive consistent Data-based evidence. However, it also results in the absence of ACK-based evidence. After expiry of a timeout period, node A determines node B to be *suspicious*, since A cannot distinguish between the case that the ACK-based evidence is lost due to link losses and the case that B launches a power control attack.
  Power control attacks are feasible only when node C is farther from node B than node A and each of the witness nodes (assuming homogeneous wireless signal propagation). Note that this attack requires precise distance calculations by node B to its neighboring nodes in order for it to adjust its transmission power appropriately.
- **Reorder** attacks are detected based on the timestamps in the received evidence. Suppose node B transmits $Data$ packet n before $Data$ packet $n-1$. As a result, each witness node overhears $Data$ packet n before $n-1$. Since $e_{n-1}.t^X > e_n.t^X$ for all evidence, S_n evaluates to false in the equation (1) and node A, therefore, decides that B is compromised.
- **Message corruption** attack is detected when node A receives evidence with different checksums. Let M′ be the message transmitted by node B in its $Data$ packet, which is different from the original message M received from node A. For every received evidence, the message checksums are not equal, i.e., $r_n \neq e_n.r_B$ and $r_n \neq e_n.r_X$, where $r_n = H[M|addr(C)]$ and $e_n.r_B = e_n.r_X = H[M'|addr(C)]$. Thus, S_n evaluates to false and node A decides that node B compromised.

B. Bypassing

In addition to launching forwarding misbehavior attacks, compromised node B can potentially disrupt our witness-based detection scheme by launching a bypassing attack as defined earlier. We explain why our witness-based detection scheme cannot be disrupted by a bypassing attack.

First, a false judge's address results in node A receiving no evidence because evidence-generating nodes cannot transmit evidence to judge A. This attack causes node A to decide that node B is compromised based on Lemma 1. Second, the compromised node B may attempt to manipulate the evidence field in a $Data$ packet so as to hide its forwarding misbehaviors. For instance, suppose that node B launches a message corruption attack manipulating the evidence field as follows:

$$B \rightarrow \{C, W\}: \quad K_B(r_B), \text{addr(A)}, \text{M}' \quad (\text{where } e_n.r_B = r_n = H[M|addr(C)])$$

Through evidence field manipulation, node B bypasses the first equality check of message checksum in equation (1). However, the evidence is still inconsistent, because $r_n \neq e_n.r_x$ if node x is not compromised. Node A decides that node B is compromised based on Lemma 2 or 3.

C. Badmouthing

Thus far, we discussed attacks launched by node B. Let us now consider the case when node C or a witness node is compromised but node B is uncompromised. In particular, we consider the scenario where node C or a witness node includes false evidence attributes (e.g., a random message checksum or a false timestamp) or transmits no evidence despite overhearing node B's correct forwarding. This may cause node A to decide that the uncompromised node B is compromised, which produces a false positive. We refer to this attack as *badmouthing*.

In data-path-based detection, node A cannot distinguish a bypassing attack by compromised node B from a badmouthing attack by compromised node C. However, as long as at least one uncompromised witness node exists, this node can produce consistent evidence of node B's correct forwarding and allow node A to distinguish that node C is compromised, as described in Algorithm 1.

Identifying a compromised node using consistent evidence also implies that the detection accuracy of the witness-based detection scheme is unaffected by multiple compromised nodes that badmouth node B, as long as there exists at least one uncompromised node $x[\in X]$. Our scheme results in false positive only if every node in the neighborhood of node B badmouths node B.

D. Collusion

Now we consider the case when compromised nodes B and $x[\in X]$ attempt to bypass the detection scheme by generating fake consistent evidence as defined earlier. For each of node B's forwarding misbehavior, we describe how nodes B and x generate fake consistent evidence to conceal node B's forwarding misbehaviors as follows:

Fake forwarding, Route deviation, and Message corruption

1) Node B launches one or more of the three forwarding misbehaviors above and transmits a $Data$ packet with a false message checksum $r_B = H[M|addr(C)] = r_n$.
2) After colluding node x receives or overhears the $Data$ packet from node B, node x includes a false message checksum $r_x = H[M|addr(C)] = r_n$, either by decrypting node B's false evidence field in the $Data$ packet or randomly guessing a message or message recipient. Finally, node x generates consistent but false

evidence, $K_x(K_B(r_B), r_x, t_x)$ where $e_n.r_B = e_n.r_x = r_n = H[M|addr(C)]$, even though node B has launched one or more of the three forwarding misbehaviors.

Reorder attack

1) Node B launches a reorder attack.
2) After colluding node x receives or overhears the $Data$ packets from node B, node x includes false timestamps for the reordered $Data$ packets to conceal node B's reorder attack, and finally generates false inconsistent evidence. This step requires that the correct sequence of $Data$ packets is known to node x.

Data-path-based detection cannot detect a collusion attack, whereas the witness-based detection scheme can expose a collusion attack as long as there is at least one inconsistent piece of evidence from an uncompromised witness node, as described in Algorithm 2. That is, our scheme produces a false negative only if every node surrounding node B is compromised and colludes with node B.

V. Performance Evaluation

In this section, we compare the detection accuracy of the data-path-based and the witness-based detection schemes in the presence of lossy links. We also quantify the communication overhead of the witness-based detection scheme and study the efficacy of the feedback suppression mechanism.

A. Metrics of detection accuracy

In our detection accuracy analysis, we individually study the two decision-making algorithms as described in section III-C: one under the assumption of collusion, and the other when the malicious nodes are assumed to act independently. The proposed model analyzes the detection accuracy as a function of the following parameters:

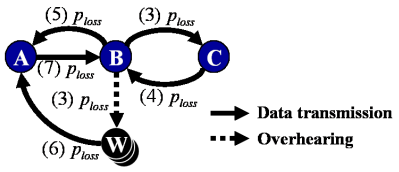


Fig. 4: **Packet loss model**: numbers in parenthesis correspond with the packet number in Fig. 3

- p_{loss} is the probability that a node fails to receive a packet from its one-hop neighbor or overhear a neighboring node's transmission. For simplicity, we assume that every node's packet loss probability is independent and identically distributed as shown in Fig. 4.
- Λ is the expected number of witness nodes located in the intersection area between node A's and node B's transmission ranges, where the number of witness nodes follows a 2D-Poisson distribution with the density parameter λ.
- p_c is the probability that a node is compromised, and is independent of p_{loss}. A compromised node launches various attacks described in section IV.
- N is the maximum number of $Data$ or ED packet retransmissions in $Data/ACK$ or ED/ED_ACK exchange.

We use false positive probability (FPP) and false negative probability (FNP) as detection accuracy metrics. For each of the non-collusion and the collusion cases, Table II illustrates the conditions under which the two algorithms result in false positives (i.e., (2), (3), (7)) and false negatives (i.e., (4), (8)). In the non-collusion case, we only observe false positives. Fake consistent evidence that may generate false negatives can be only created through collusion, as explained earlier (i.e., FNP is equal to 0 in the non-collusion case). In Table IIa, we divide the occurrence of false positives in the non-collusion case into two disjoint events as follows:

- Node A receives no evidence when node B is not compromised (labeled (2)).
- Node A only receives inconsistent evidence when node B is not compromised (labeled (3)).

In Appendix, we provide analytic expressions that quantify each of the FPPs and FNPs for both data-path-based and witness-based detection. As an example, assuming non-collusion, the probabilities of false positive event due to (2) in Table IIa for the data-path-based detection ($P[FP^{(2)}_{d,nc}]$) and the witness-based detection ($P[FP^{(2)}_{w,nc}]$) is given respectively by:

$$\begin{aligned} P[FP^{(2)}_{d,nc}] &= 1 - \left(1 - (1 - (1 - p_{loss})^2)^N\right) \cdot \left(1 - p_{loss}^N\right) \\ P[FP^{(2)}_{w,nc}] &= \left(1 - \left(1 - (1 - (1 - p_{loss})^2)^N\right) \cdot \left(1 - p_{loss}^N\right)\right) \\ &\quad \times exp(-\Lambda(1 - p_{loss}^N)^2) \end{aligned}$$

$P[FP^{(2)}_{d,nc}]$ computes the probability that node C's ACK-based evidence fails to reach node A via uncompromised node B. $P[FP^{(2)}_{w,nc}]$ is the probability that neither node C nor witness nodes succeed in transmitting evidence to node A, when node B is not compromised. The first term of $P[FP^{(2)}_{w,nc}]$ is equal to $P[FP^{(2)}_{d,nc}]$, and the second term calculates witness nodes' failure of evidence transmission to node A. The probabilities of other events in the Table II are calculated similarly.

B. Numerical evaluation of detection accuracy

We now observe how the FPP and FNP vary with the expected number of witness nodes and packet loss probability in both the non-collusion and the collusion cases. Note that the case of data-path-based detection corresponds to the case of $\Lambda = 0$. In our results below, we assume that a packet can be re-transmitted up to three times ($N = 3$); once this maximum number of retransmission is reached, the packet is considered lost (dropped) by the sender and is not received at the receiver.

- **FPP in the non-collusion case**: Fig. 5 plots the FPP as a function of the expected number of witness nodes (Λ) for $p_{loss} = 0.1, 0.5, 0.9$, and $p_c = 0.1$ (Fig. 5a) and $p_c = 0.5$ (Fig. 5b). As expected, the FPP decreases (improving detection accuracy) as Λ increases, since an increased number of witness nodes improves success probability of overhearing and reliability of a path to node A, thus

(a) Non-collusion case

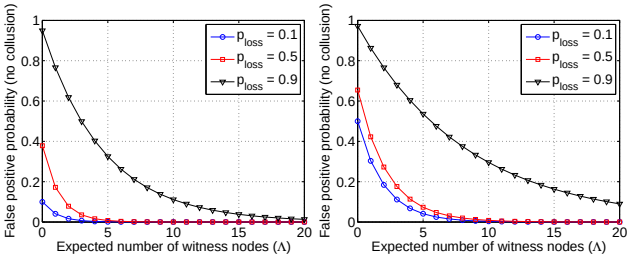
Evidence received by Node A	Node A's decision	Node B's actual state	
		Uncompromised	Compromised
At least one evidence is consistent	Uncompromised	(1) True positive	(4) False negative
No evidence	Compromised	(2) False positive	(5) True negative
All evidence is inconsistent		(3) False positive	

(b) Collusion case

Evidence received by Node A	Node A's decision	Actual presence of an attack among B, C, W	
		Nonexistent	Existent
All evidence is consistent	Nonexistent	(6) True positive	(8) False negative
No evidence	Existent	(7) False positive	(9) True negative
At least one evidence is inconsistent		-	

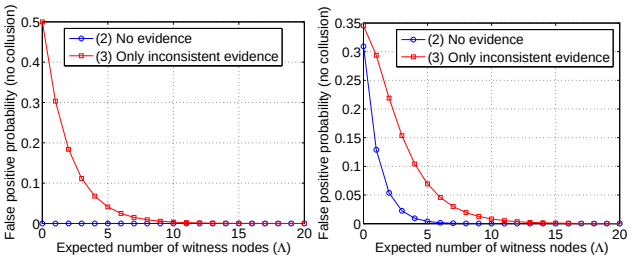
TABLE II: Occurrence of False positives and False Negatives

provides increased evidence to the judge node. The FPP also decreases as p_{loss} decreases, for the same reason. Fig. 6 breaks down the causes of the false positives for the case of $p_c = 0.5$. When p_{loss} is small (Fig. 6a) the source of false positives is primarily badmouthing (event (3) in Table IIa), while the FPP due to lack of evidence (event (2) in Table IIa) is almost zero. When p_{loss} increases to 0.5 in Fig. 6b, the FPP resulting from a lack of evidence increases.



(a) $p_c = 0.1$ (b) $p_c = 0.5$

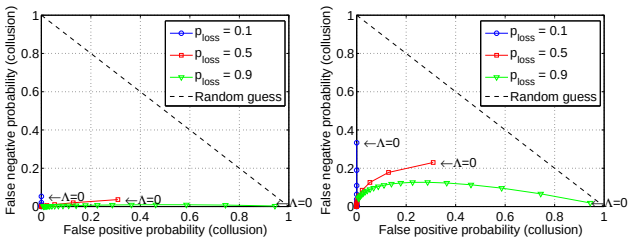
Fig. 5: **FPP in the no-collusion case**



(a) $p_{loss} = 0.1$ (b) $p_{loss} = 0.5$

Fig. 6: **Breakdown of the causes of false positives in the no-collusion, where** $p_c = 0.5$

- **FPP and FNP in the collusion case**: Fig. 7 plots the FPP versus the FNP for $p_c = 0.1$ (Fig. 7a) and $p_c = 0.5$ (Fig. 7b). Each curve corresponds to a different packet loss probability (p_{loss}), and each point on a curve corresponds to a different expected number of witness



(a) $p_c = 0.1$ (b) $p_c = 0.5$

Fig. 7: **FPP and FNP in the collusion case**

nodes (Λ). The upper rightmost point on each curve corresponds to $\Lambda = 0$ (data-path-based detection). As Λ increases, both the false positive and false negatives generally decrease, demonstrating the overall value of using a witness-based approach. The once exception is in the case of extremely high packet loss probability ($p_{loss} = 0.9$). False negatives slowly increase as Λ increase, unless witness density is high enough to successfully receive inconsistent evidence, which exposes a collusion attack.

C. Communication overhead

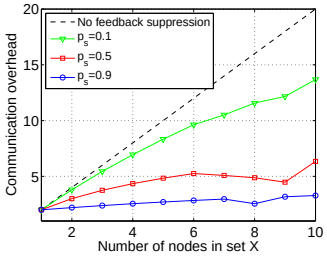


Fig. 8: **Communication overhead (when node** $x[\in X]$ **does not retransmit** ED **packets)**

We observe the reduced number of redundant ED and ED_ACK packets by feedback suppression: Note that since feedback suppression is based on an ED_ACK packet from node A, it does not decrease the probability of evidence being successfully transmitted in the witness-based detection

scheme. We define communication overhead as the expected number of ED and ED_ACK packets. Each solid curve in Fig. 8 plots the communication overhead for different feedback suppression success probabilities (p_s). p_s is the probability that a node receives or overhears an ED_ACK packet. Derivations can be found in Appendix. A dotted curve denotes the communication overhead without feedback suppression. Fig. 8 shows that feedback suppression results in relatively low communication overhead, almost independent of the number of nodes in X over a wide range of values of p_s.

VI. Related Work

There have been numerous proposals related to the problem of detecting forwarding misbehavior in wireless networks. We divide these proposals into two broad categories, based on whether evidence of forwarding behavior is gathered passively or actively. Approaches based on passive measurement typically involve a node in promiscuous mode overhearing and monitoring its neighboring node's data transmissions, while the active measurement based approaches inject probe packets along a data path and infer the state of forwarding nodes on a data path from received probe responses.

Neighborhood watch schemes employing passive measurements have been proposed in [2, 5, 12, 13, 15, 17, 21]. In these proposals, each node typically monitors its neighbor node's forwarding behavior by matching the neighbor node's incoming and outgoing data packet pair. However, this detection metric can cause a high rate of false positives, since mismatched incoming and outgoing data packets can often occur in wireless network due to interference and channel errors. Some of these schemes reduce the false positive rates, by deriving mismatch rate of a series of incoming and outgoing data packet pairs. Other schemes also attempt to precisely decide forwarding nodes' states by sharing observations with multiple collaborating nodes.

Both the accumulative (observing a series of packets) and the collaborative observation based scheme have their own limitations. Solutions relying on the accumulative scheme have an inevitable detection delay before detecting a misbehavior. Schemes relying on observations from multiple collaborating nodes remove the detection delays associated with the accumulative scheme. However, the authenticity of the observations from neighbors cannot be guaranteed. In particular, if a node becomes compromised, an attacker can send out false information signed using the stolen key, easily thwarting these schemes. These collaborative schemes can also cause high communication overhead.

There are several works employing active measurements [1, 7, 9, 10, 16] that involve a node sending a probe packet to its observing node. Absence of a response from the observing node within a predefined time is used as evidence to incriminate an intermediate node. However, these schemes allow an attacker to generate valid probe responses using a stolen key without correctly forwarding data packet.

Routing protocols [3, 4, 8, 18] support link breakage detection mechanisms, which may be adapted for drop attack detection. However, the link breakage mechanisms can only detect drop attacks in the control plane i.e., when an adversary does not transmit periodic *Keepalive* packets. If an adversary only transmits the *Keepalive* packets but no data packets, a link breakage mechanism cannot handle data packet dropping, which is the primary focus of this paper.

VII. Conclusion

In this paper, we presented witness-based detection, a new scheme that verifies correct forwarding along data paths in wireless networks. Using observations from multiple witness nodes, our scheme either identifies a source of forwarding misbehavior (when there is no collusion among nodes) or exposes the presence of a misbehaving node (when malicious nodes collude with one another). Unlike existing schemes, the witness-based detection scheme detects misbehaving nodes without incurring significant delays, even in the presence of multiple adversaries. Under the assumption of reliable communication between nodes, we formally showed that the witness-based detection scheme does not produce false positives or false negatives, as long as there is at least one uncompromised witness node. Using an analytical model, we studied the performance of our scheme in a realistic wireless setting. Our analysis showed that witness-based detection can support low false positive and false negative rates even in the presence of highly lossy wireless links, without incurring a significant communication overhead.

Acknowledgment

This material is based upon work under a subcontract #069153 issued by BAE Systems National Security Solutions, Inc. and supported by the Defense Advanced Research Projects Agency (DARPA) and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego under Contract No. N66001-08-C-2013.

References

[1] B. Awerbuch et al., "An On-Demand Secure Routing Protocol Resilient to Byzantine Failures," in Proc. ACM workshop on Wireless security, 2002, pp.21-30.

[2] S. Buchegger and JY. Le Boudec, "Performance Analysis of the CONFIDANT Protocol," in Proc. ACM international symposium on MobiHoc, June 2002, pp. 226-236.

[3] T. Clausen, P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," IETF Network Working Group RFC3626, Oct 2003.

[4] R. Dube, C. D. Rais , Kuang-Yeh Wang , S. K. Tripathi, "Signal Stability based Adaptive Routing (SSA) for Ad-Hoc Mobile Networks," IEEE Personal Communications, vol.4, no.1, Feb 1997, pp.36-45.

[5] Q. He, D. Wu, and P. Khosla, "SORI: a secure and objective reputation-based incentive scheme for ad-hoc networks," in Proc. IEEE WCNC, 2004, pp. 21-25.

[6] YC. Hu, A. Perrig, "A survey of secure wireless ad hoc routing," IEEE Security & Privacy, vol.2, no.3, May.2004, pp.28-39.

[7] Q. Huan, I. C. Avramopoulos, H. Kobayashi, and B. Liu, "Secure data forwarding in wireless ad hoc networks," in Proc. IEEE international conference on communications, 2005, pp.3525-3531.
[8] D. Johnson, Y. Hu, and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," IETF Network Working Group RFC4728, Feb 2007.
[9] M. Just, E. Kranakis, T. Wan, "Resisting Malicious Packet Dropping in Wireless Ad-Hoc Networks," in Proc. 2nd annual Conference on Adhoc Networks and Wireless, 2003, pp.151-163.
[10] F. Kargl et al., "Advanced detection of selfish or compromised nodes in ad hoc networks," in 1st European Workshop Security in Ad-Hoc and Sensor Networks, Aug.2004.
[11] N. Komninosa, D. Vergadosa, and C. Douligeris, "Detecting unauthorized and compromised nodes in mobile ad hoc networks," Ad Hoc Networks, Elsevier, vol.4, no.3, April 2007, pp. 289-298.
[12] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in Proc. ACM international conference on MobiCom, 2000, pp. 255-265.
[13] D. McCoy, D Sicker, and D. Grunwald, "A Mechanism for Detecting and Responding to Misbehaving Nodes in Wireless Networks," in Proc. IEEE Communications Society Conference on SECON, 2007, pp.678-684.
[14] A. D. McDonald, J. Crowcroft and M. Srivatsa, "Security across Disparate Management Domains in wireless networkss,in Proc. annual conference of ITA, 2008. pp. 123-130.
[15] P. Michiardi and R. Molva, "Core: A COllaborative REputation Mechanism to enforce node cooperation in Mobile Ad Hoc Networks," in Proc. IFIP TC6/TC11 Sixth Joint Working Conference on Communications and Multimedia Security, 2002, pp. 107-121.
[16] V. N. Padmanabhan and D. R. Simon. "Secure Traceroute to Detect Faulty or Malicious Routing," in Proc. 1st Workshop on Hot Topics in Networks, 2002, pp.77-82.
[17] K. Paul and D. Westho, "Context aware inferencing to rate a selfish node in DSR based ad-hoc networks," in Proc. IEEE Globecom Conference, 2002, pp. 178-182.
[18] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," IETF Network Working Group RFC3561, July 2003.
[19] R. Ramanujan, S. Kudige, S. Takkella, T. Nguyen, and F. Adelstein, "Intrusion-resistant ad hoc wireless networks," in Proc. IEEE MILCOM, 2002, pp. 890-894.
[20] D. Sterne, P. Balasubramanyam, D. Carman, B. Wilson, R. Talpade, C. Ko, R. Balupari, C.-Y. Tseng, and T. Bowen, "A general cooperative intrusion detection architecture for MANETs,", in Proc. IEEE Workshop on Information Assurance, 2005, pp. 57-70.
[21] H. Yang, J. Shu, X. Meng, and S. Lu, "SCAN: Self-organized Network Layer Security in Mobile Ad Hoc Networks," IEEE Journal on Selected Areas in Communications, Special Issue on Security in Wireless Ad Hoc Networks, vol.24, no.2, Feb.2006, pp.261-273.

APPENDIX

A. Packet Loss Model

We introduce a packet loss model that is used in our analysis of detection accuracy of the data-path-based and the witness-based detection schemes.

1) Assumption: We make the following assumptions in our model:

- Each node on data path A-B-C and witness nodes near the data path are assumed to be static for the duration of path verification.
- To restrict our attention to witness nodes' Data-based evidence dissemination, we only consider witness nodes that are located within the transmission ranges of both nodes A and B (we refer to this area as *witness area* and Δ denotes the witness area). Thus, witness nodes in witness area (Δ) can overhear B's $Data$ transmission and directly transmit Data-based evidence back to node A.
- Witness nodes are spatially distributed according to a 2D Poisson with intensity λ.

2) Model description: On basis of the definitions of four parameters in Section V, we supplement their descriptions.

- p_{loss} is the probability that a node fails to receive a packet from its one-hop neighbor or overhear a neighboring node's transmission. For simplicity, we assume that every node's packet loss probability is independent and identically distributed as shown in Fig. 4. That is, we do not differentiate overhearing from data transmission and assume that both data transmission and overhearing experience equivalent interference.
- p_c is the probability that a node is compromised, and p_c is independent of the probability p_{loss}. In other words, we separate packet losses occurring due to interference and channel errors, and those caused by a compromised node launching a drop or power control attack, or evidence drop.
- Λ is the expected number of witness nodes in witness area Δ, where $\Lambda = \Delta \times \lambda$.
- N is the maximum number of $Data$ or ED packet retransmissions in $Data/ACK$ or ED/ED_ACK exchange.

Before we derive the detection accuracy metrics, we compute two quantities that will be used extensively in our detection accuracy metrics.

- $P[\neg C_e]$ is the probability that node A fails to receive ACK-based evidence, originally from node C, via node B. The event $\neg C_e$ occurs in each of the following cases:
  1) Node B fails to receive an ACK or ED packet (conveying ACK-based evidence) from node C even after node B retransmits a $Data$ packet up to maximum N times. In this case, node C may receive

i $(0 \leq i \leq N)$ out of node B's N $Data$ packets, and replies with i ACK packets, but node B fails to receive any of i ACK packets. Let $\neg B_e$ denote this case. The probability of $\neg B_e$ can be computed as follows:

$$\begin{aligned} P[\neg B_e] &= \sum_{i=0}^{N} \left(\binom{N}{i} p_{loss}^{N-i} (1-p_{loss})^i \right) \cdot p_{loss}^i \\ &= (1-(1-p_{loss})^2)^N \end{aligned}$$

2) Node A fails to receive an ED packet (conveying ACK-based evidence) relayed by node B after node B successfully receives an ACK or ED packet from node C. The probability of this case is given as follows:

$$P[\neg C_e | B_e] = p_{loss}^N$$

From the law of total probability, we have:

$$P[\neg C_e] = P[\neg C_e | \neg B_e] P[\neg B_e] + P[\neg C_e | B_e] P[B_e]$$

But $P[\neg C_e | \neg B_e] = 1$. Therefore,

$$P[\neg C_e] = P[\neg B_e] + P[\neg C_e | B_e] P[B_e]$$

Substituting for $P[\neg B_e]$, we get

$$P[\neg C_e] = (1-(1-p_{loss})^2)^N + p_{loss}^N (1-(1-(1-p_{loss})^2)^N)$$

- $P[\neg W_e^i]$ is the probability that none of the witness nodes in set W^i $(i = 0, 1)$ succeed in evidence dissemination, including the case when there are no witness nodes within witness area (Δ). W^0 denotes the set of uncompromised witness nodes, while W^1 is the set of compromised witness nodes. $W = W^0 \cup W^1$ and $W^0 \cap W^1 = \emptyset$. This event occurs when either: 1) none of the witness nodes in set W^i overhear $Data$ packets, or 2) all the witness nodes that have overheard $Data$ packets fail in transmitting Data-based evidence back to node A.

$$\begin{aligned} P[\neg W_e^0] &= exp\left(-\Lambda(1-p_c)(1-p_{loss}^N)^2\right) \\ P[\neg W_e^1] &= exp\left(-\Lambda p_c(1-p_{loss}^N)^2\right) \\ P[\neg W_e] &= exp\left(-\Lambda(1-p_{loss}^N)^2\right) \end{aligned}$$

$$P[\neg W_e^i] = \sum_{n=0}^{\infty} P[\Lambda_n] \left(\sum_{m=0}^{n} C(m,n,i) P[\Omega_m] \right)$$

  - n is the number of witness nodes in witness area. In that, $|W| = n$.
  - $P[\Lambda_n]$ is the probability that there are n witness nodes in witness area.

$$P[\Lambda_n] = \frac{\Lambda^n}{n!} exp(-\Lambda), \text{ where } n \geq 0$$

  - $C(m,n,i)$ is the probability that m out of the n witness nodes are uncompromised $(i = 0)$ or compromised $(i = 1)$.

$$\begin{aligned} C(m,n,0) &= \binom{n}{m} (1-p_c)^m p_c^{(n-m)} \\ C(m,n,1) &= \binom{n}{m} p_c^m (1-p_c)^{(n-m)} \end{aligned}$$

  - $P[\Omega_m]$ is the probability that m uncompromised (or compromised) witness nodes fail in evidence dissemination to judge node A.

$$\begin{aligned} P[\Omega_m] =& \sum_{k=0}^{m} P[k \text{ out of the } m \text{ witness nodes overhear } Data \text{ packets}] \\ & \cdot P[\text{All evidence from } k \text{ witness nodes are lost}] \\ =& \sum_{k=0}^{m} \left(\binom{m}{k} p_{loss}^{N(m-k)} (1-p_{loss}^N)^k \right) \cdot (p_{loss}^N)^k \\ =& (2p_{loss}^N - p_{loss}^{2N})^m \end{aligned}$$

B. Detection Accuracy Metrics

Using the packet loss model described, we derive detection accuracy metrics for the two algorithms proposed in Section III: false positive probability (FPP), false negative probability (FNP). Let $FP_{d,nc}$ (and $FN_{d,nc}$) denote the event that a false positive occurs (false negative respectively) in the non-collusion case for the data-path-based detection scheme. Let $FP_{d,c}$ and $FN_{d,c}$ denote the same events for the data-path-based detection scheme, when compromised nodes are assumed to collude with each other. Similarly, let $FP_{w,nc}$, $FP_{w,c}$, $FN_{w,nc}$ and $FN_{w,c}$ denote the respective metrics for the witness-based detection scheme. Let Z_x be a boolean variable that is 1 when node $x[\in X]$ is compromised, and 0 otherwise.

- **FPP in the non-collusion case** is the probability that node A does not receive consistent evidence when node B is not compromised. A false positive event results from one of the following two cases:
  - Case (2) in TABLE II.a denotes the case when no evidence is transmitted to node A due to wireless link losses. Node A incorrectly decides node B to be compromised, as per Lemma 1.
    - $P[FP_{d,nc}^{(2)}] = P[\neg C_e]$.
    - $P[FP_{w,nc}^{(2)}] = P[\neg C_e] P[\neg W_e]$.
  - Case (3) in TABLE II.a is caused when every evidence received by node A is inconsistent. Note that this case allows for the existence of an uncompromised node, but the uncompromised nodes' evidence does not reach node A due to wireless losses. Node A incorrectly determines B to be compromised, as per Lemma 2.
    - $P[FP_{d,nc}^{(3)}] = P[Z_C = 1] P[C_e]$.
    - $P[FP_{w,nc}^{(3)}]$ is obtained by conditioning on node C's state (uncompromised, compromised) as follows:

$$P[FP^{(3)}_{w,nc}] = P[Z_C = 0]P[\neg C_e]P[\neg W^0_e]P[W^1_e] + P[Z_C = 1]P[\neg W^0_e](1 - P[\neg C_e]P[\neg W^1_e])$$

- **FNP in the non-collusion case** is the probability that node A only receives consistent evidence and node B is compromised. In our proposed evidence format described in Section III, it is easy to see that a compromised node cannot generate consistent evidence without colluding with another compromised node. Thus,

$$P[FN_{d,nc}] = P[FN_{w,nc}] = 0$$

- **FPP in the collusion case** is the probability that node A does not receive consistent evidence even though none of nodes B, C, and set W are compromised (listed as the case (7) in TABLE II). Unlike the non-collusion case, Algorithm 2 does not yield a false positive due to inconsistent evidence. FPP for each of the detection schemes under collusion is given as follows:
  - $P[FP_{d,c}] = P[\neg C_e]$.
  - $P[FP_{w,c}] = P[\neg C_e]P[\neg W_e]$.
- **FNP in the collusion case** is the probability that every evidence received by node A is consistent, even though there exists at least one compromised node among nodes B, C, and set W (listed as the case (8) of TABLE II). Note that we assume that node B and node $x[\in X]$ collude with each other if once nodes B and x are compromised. In other words, we do not consider that compromised nodes B and x independently launch attacks described in Section IV, which generate inconsistent evidence. Before deriving an equation for FNP, we illustrate five conditions that cause a false negative:
  1) There is at least one compromised node among nodes B, C, and set W.
  2) Node B is compromised, and launches forwarding misbehaviors with a bypassing-attack, which puts a false message checksum.
  3) There exists at least one compromised node x in set X.
  4) Fake consistent evidence produced by node B in collusion with node x successfully reaches node A.
  5) However, every inconsistent evidence, generated by an uncompromised node in set X for exposing node B's forwarding misbehaviors, fails to reach node A.

  Thus, in the data-path-based detection scheme, FNP ($P[FN_{d,c}]$) stands for the probability that fake consistent evidence generated by collusion between nodes B and C successfully reaches node A, when there is at least one compromised node between nodes B and C. The FNP in the data-path-based detection scheme is given as follows:

$$P[FN_{d,c}] = \frac{P[Z_B = 1, Z_C = 1]}{1 - P[Z_B = 0, Z_C = 0]} P[C_e]$$

  In the witness-based detection scheme, we compute FNP ($P[FN_{w,c}]$) by dividing a false negative into two events depending on node C's state, considering witness nodes in witness area. $P[FN^1_{w,c}(n)]$ is the FNP when node C is not compromised. $P[FN^2_{w,c}(n)]$ is the FNP when node C is compromised. Thus, $P[FN^1_{w,c}(n)]$ conditions the probability of event $\neg C_e$, but $P[FN^2_{w,c}(n)]$ does the probability of event C_e.

$$\begin{aligned} P[FN_{w,c}] &= \sum_{n=0}^{\infty} \left(P[FN^1_{w,c}(n)] + P[FN^2_{w,c}(n)]\right) \\ P[FN^1_{w,c}(n)] &= \left(\frac{P[Z_B = 1, Z_C = 0]}{1 - P[Z_B = 0, Z_C = 0, |W^1| = 0]}\right) \\ &\quad \cdot P[\neg C_e]P[\neg W^0_e(n)]P[W^1_e(n)] \\ P[FN^2_{w,c}(n)] &= \left(\frac{P[Z_B = 1, Z_C = 1]}{1 - P[Z_B = 0, Z_C = 0, |W^1| = 0]}\right) \\ &\quad \cdot P[\neg W^0_e(n)](1 - P[\neg C_e]P[\neg W^1_e(n)]) \end{aligned}$$

C. Communication Overhead

We describe how we derive communication overhead of the witness-based detection scheme in Section V. For focusing on the reduced amount of communication overhead by feedback suppression, we assume the following:

- All nodes in set X have overheard $Data$ packets during the evidence-generation step, and attempt to transmit evidence to node A.
- ED/ED_ACK exchange between a node (i.e., an ED packet transmitter) in set X and node A does not fail.

Before illustrating an equation for communication overhead, we first define variables and a probability constituting the equation:

- p_s is the probability that a node in set X receives or overhears an ED_ACK packet.
- ρ is a random variable on the number of ED/ED_ACK exchanges (k), until all nodes in set X receive or overhear ED_ACK packets. By the assumption that an ED/ED_ACK exchange does not fail, $1 \leq k \leq |X|$.
- x_i $(i = 0, 1, \cdots, k)$ is the number of nodes (excluding an ED packet transmitter) that fail to overhear an ED_ACK packet for the i-th ED/ED_ACK exchange, where $x_0 = |X|$. Let $\overline{x} = \{x_0, \cdots, x_k\}$.
- $f_i(\overline{x}, p_s)$ is the probability that x_i out of x_{i-1}-1 nodes fail to overhear an ED_ACK packet for the i-th ED/ED_ACK exchange. Here, we exclude one node as an ED packet transmitter among nodes that have failed to overhear an ED_ACK packet previously.

$$f_i(\overline{x}, p_s) = \binom{x_{i-1} - 1}{x_i} (1 - p_s)^{x_i} p_s^{x_{i-1} - x_i - 1}$$

Thus, the probability mass function on random variable ρ given p_s, $P[\rho = k|p_s]$, and its expectation are computed as follows:

$$P[\rho = k|p_s] = \sum_{\overline{x}} \left(\prod_{i=1}^{k} f_i(\overline{x}, p_s) \right)$$

$$E[\rho|p_s] = \sum_{k=1}^{|X|} (k \cdot P[\rho = k|p_s])$$

Finally, we derive the expected number of ED and ED_ACK packets for each message's verification by $2 \times E[\rho|p_s]$.