Epidemic Attacks in Network-Coding Enabled Wireless Mesh Networks: Detection, Identification and Evaluation

Yongkun Li and John C.S. Lui, *Fellow, IEEE, Fellow, ACM*

Abstract—Epidemic attack is a severe security problem in network-coding enabled wireless mesh networks (WMNs). Malicious nodes can easily launch such form of attack to create an epidemic spreading of polluted packets and deplete network resources. The contribution of this work is to address such security problem. We allow the presence of "smart" attackers, i.e., they can pretend to be legitimate nodes to probabilistically transmit valid packets so as to reduce the chance of being detected. We also address the case where attackers cooperatively inject polluted packets. We employ the time-based checksum and batch verification to determine the existence of polluted packets, then propose a set of fully "distributed" and "randomized" detection algorithms so that each legitimate node in a WMN can identify its malicious neighbors and purge them for future communication. We provide formal analysis to quantify the performance of the algorithms. Furthermore, simulation and system prototyping are carried out to validate the theoretic analysis and show the effectiveness and efficiency of the detection algorithms.

Index Terms—Pollution Attack, Wireless Mesh Networks, Network Coding, Performance Evaluation

1 INTRODUCTION

In recent years, wireless mesh networks (WMNs) have emerged as a promising platform to provide easy Internet access [1], [3], [16]. However, due to the spatial and temporal fading of the wireless channels, communication links between nodes usually have high loss rates. As reported in [1], half of the operational links have a loss probability greater than 30%. Therefore, traditional routing protocols, which determine the next hop in forwarding a packet, cannot guarantee a high end-to-end throughput. In the past few years, we have seen some exciting advancements in wireless routing protocols to improve the performance of WMNs [4], [17], [22]. One such protocol that is receiving a lot of attention is the *opportunistic routing protocol* [5], [10]. In this protocol, any node which overhears the transmission of another node can participate in packet forwarding. Using this paradigm, high end-to-end throughput can be obtained even if some links along the source-destination path are lossy. However, since multiple nodes which overhear the packet can participate in the packet forwarding, packet collision may occur and thereby reduces the network capacity. To overcome this problem, a number of researchers have proposed enhancement on the transmission schedulers. The main idea is that a node which is *closer* to the destination will transmit with a higher priority so as to reduce the possibility of packet collision.

To further improve the spatial reuse, a transmission scheduler based on network coding [20], [21] was proposed. This promising approach can not only improve the end-to-end throughput, but also reduce packet collision and improve the network capacity. As demonstrated in [19] and systems like COPE [13] and MORE [6], one can achieve the above claim for unicast and multicast data delivery in WMNs. The core idea of using network coding is in "*packets mixing*": intermediate nodes along the source-destination path can mix (or encode) received packets and then forward the coded packet to other nodes. This active mixing by intermediate nodes increases the packet diversity, resulting in fewer redundant packet transmissions, and improves network capacity. As long as the destination receives enough innovative (or linearly independent) packets, it can decode the received packets and obtain the original data.

However, allowing nodes in a WMN to perform network coding opens the door for *epidemic (or pollution) attack*. Malicious nodes can inject polluted/bogus packets into the wireless network. If an intermediate node is unaware of receiving a polluted packet, it will continue to perform the packet encoding and then forward the encoded but corrupted packet to its neighbors. Since all nodes in a WMN participate in encoding and packet forwarding, polluted packets will behave like an epidemic and can be easily propagated across the entire network, thereby significantly consume the network resource and degrade the performance of legitimate flows. As indicated in [8], pollution attack can be easily launched, and some related work, e.g., [11], [14], [18], [23], [24] addresses this problem, in particular, on detecting the existence of pollution attack in the network and how to discard polluted packets.

In this paper, we focus on "*detecting*" and "*identifying*" epidemic attackers in WMNs. Attackers can be *intelligent* in the sense that they can choose to forward corrupted encoded packets, or they can choose to forward legitimate encoded packets. The rationale for attackers to *pretend* as legitimate nodes is to reduce the chance of being detected. Contributions of our work are:

- We propose a *randomized* and *fully distributed* detection

• *The authors are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong. E-mail: ykli@cse.cuhk.edu.hk, cslui@cse.cuhk.edu.hk*

gital Object Identifier 10.1109/TMC.2012.186 1536-1233/12/$31.00 © 2012 IEEE

mechanism: any legitimate node in a WMN can execute our detection algorithms to identify its malicious neighbors. We allow malicious nodes to pretend as legitimate nodes and cooperatively inject polluted packets.
- We present a general analytical framework to quantify the performance of our detection algorithms which shows the effectiveness and efficiency of the algorithms.
- We validate our analytical models via extensive simulations and system prototype. The simulation results show the accuracy of our models and the effectiveness and the efficiency of the algorithms.

The outline of the paper is as follows. In Section 2, we briefly provide the necessary background on network coding and how to efficiently determine the existence of polluted packets. In Section 3, we present the distributed detection mechanism in detail and also provide analysis on its performance measures. In Section 4, we present a generalized detection algorithm which cannot only defend against cooperative attack, but also speed up the detection. We validate our analysis via simulations and experimental results in Section 5, and Section 6 concludes.

2 Network Coding and Time-based Checksum Batch Verification

In this section, we provide a brief background on network-coding enabled WMNs, as well as the mechanism of time-based checksum batch verification which is used to determine the existence of polluted packets.

Wireless mesh networks (WMNs) consist of two types of nodes: mesh routers and mesh clients. Each node operates not only as a host but also as a router which forwards packets for other nodes that are not in the direct transmission range of their destinations. Although mesh clients can be stationary or mobile, mesh routers usually have minimal mobility. For the most commonly used architecture of WMNs, there is a backbone network which only consists of mesh routers. In this paper, we assume that the mesh routers are stationary, and we focus on the backbones of WMNs which configured network-coding enabled opportunistic routing protocol. This assumption is not restrictive in practical WMNs as the network-coding enabled opportunistic routing protocol cannot only decrease the packets loss rate but also reduce the packets collision so as to improve the network throughput.

Let us provide a brief background on network coding. The pioneering paper on network coding can be found in [2]. Authors in [21] show that linear network coding achieves the maximum capacity bounds. Moreover, Ho et al. prove that the above argument is also true even if random coefficients [12] are used for coding. When network coding is applied in WMNs, the source first breaks up the file or message into multiple generations. Each generation is further divided into n packets, which are usually referred to as *native packets*. Each packet is further divided into m codewords, each of which is regarded as an element in a finite field $\mathbb{F}_q$, where q is a positive power of a prime number. Each native packet $\overrightarrow{p_i}$ can be viewed as an m-dimensional vector over the field $\mathbb{F}_q$, i.e.,

$$\overrightarrow{p_i} = (p_{1i}, p_{2i}, ..., p_{mi})^\top, \qquad p_{ji} \in \mathbb{F}_q.$$

So all packets in one generation can be denoted as a $m \times n$ matrix G, i.e., $G = [\overrightarrow{p_1}, \overrightarrow{p_2}, ..., \overrightarrow{p_n}]$. When the 802.11 MAC is ready to send a packet, the source creates a random linear combination of the n native packets and then transmits the coded packet. Formally, a *coded packet* is $(\overrightarrow{e_j}, \overrightarrow{c_j})$ where $\overrightarrow{e_j} = \sum_{i=1}^{n} c_{ji}\overrightarrow{p_i}$, and c_{ji} is a random coefficient. We call $\overrightarrow{c_j} = (c_{j1}, c_{j2}, ..., c_{jn})$ the code vector. For a forwarder, if it receives l coded packets of the form $(\overrightarrow{e_j}, \overrightarrow{c_j})$, then it also creates a linear combination of the coded packets via $\overrightarrow{e} = \sum_{j=1}^{l} c'_j\overrightarrow{e_j}$, where c'_j is a random coefficient. Note that, a linear combination of coded packets is also a linear combination of the native packets. Specifically, $\overrightarrow{e} = \sum_{j=1}^{l} c'_j\overrightarrow{e_j} = \sum_{i=1}^{n}(\sum_{j=1}^{l} c'_j c_{ji})\overrightarrow{p_i}$, which is also expressed as a linear combination of the native packets.

When network coding is applied, intermediate malicious nodes can modify the received packets and forward polluted/bogus packets. We say a packet $(\overrightarrow{e_j}, \overrightarrow{c_j})$ is valid/correct if and only if $\overrightarrow{e_j} = \sum_{i=1}^{n} c_{ji}\overrightarrow{p_i}$ holds. Otherwise, we call it a polluted/bogus packet. Note that, the possibility of packets mixing using network coding makes WMNs vulnerable to pollution attack, which is induced by malicious nodes injecting polluted packets. A naive approach to defend against such attack is to perform hash verification by using hash functions [15], which can check the integrity of each coded packet. However, due to the high computational cost of modular exponentiation, the achieved throughput is very low even if batch verification is employed. One practical scheme to verify packets is the time-based checksum verification mechanism [8], which only requires random linear transformation to reduce the computational complexity. The idea of time-based checksum verification is as follows. The source periodically broadcasts checksum packets which include $(CHK_s(G), s, t)$ in an authenticated way after broadcasting multiple data packets, where s is a random κ-bit seed and $CHK_s(G)$ is a checksum which is a $b \times n$ matrix and t is the time when the checksum is generated. To generate the checksum $CHK_s(G)$, the source first generates a random $b \times m$ matrix H_s by using the random κ-bit seed s and a pseudo-random function $f : \{0,1\}^\kappa \times \{0,1\}^{\log_2 b + \log_2 m} \rightarrow \mathbb{F}_q$. Specifically, the element $(H_s)_{ij}$ is generated via $f_s(i||j)$. After generating the matrix H_s, the source generates the checksum $CHK_s(G)$ for generation G via $CHK_s(G) = H_s G$.

When a node receives a checksum packet, it verifies the coded packets which are received before the checksum is generated, i.e., before time $t - \Delta$ where Δ is the maximum time skew in the network. Formally, to verify a coded packet $(\overrightarrow{e_j}, \overrightarrow{c_j})$, it checks if the following equation holds.

$$CHK_s(G)\overrightarrow{c_j} = H_s\overrightarrow{e_j}, \tag{1}$$

where H_s is generated from the seed s by suing the way described before. If Equation (1) holds, then the coded packet is valid, otherwise, we call it a polluted packet. To reduce the computational cost and make the verification practical, the node can not verify every received packet, but perform *batch verification* instead. Based on the batch verification, when a node receives a checksum packet and has received l coded packets $(\overrightarrow{e_j}, \overrightarrow{c_j})$ $(j = 1, 2, ..., l)$, it first creates a linear combination of these l packets, $\overrightarrow{e} = \sum_{j=1}^{l} c'_j\overrightarrow{e_j} =$

$\sum_{i=1}^{n}(\sum_{j=1}^{l} c'_j c_{ji})\overrightarrow{p_i}$, then verifies the the coded packet $(\overrightarrow{e}, \overrightarrow{c})$ where $\overrightarrow{c} = \sum_{j=1}^{l} c'_j \overrightarrow{c_j}$ by checking whether the following equation holds or not.

$$CHK_s(G)\overrightarrow{c} = H_s\overrightarrow{e}. \quad (2)$$

One important note is that using the batch verification, if Equation (2) holds, then it implies all l received packets are valid, and we call it the batch verification matches. On the other hand, if the equality does not hold, then at least one of the l packets is polluted and we have *no knowledge* of which are the polluted packets. In this case, we call it the batch verification does not match. In this paper, we assume that whenever a verification is needed, batch verification is performed. This assumption is not restrictive, in particular, when one considers designing a WMN with high throughput and low end-to-end delay.

Due to the packets mixing property of network coding, i.e., any malicious node can easily modify the packets, WMNs are prone to pollution attack. As shown in [9], the threat of pollution attack is very severe. Comparing with previous work which focuses on designing efficient packet verification scheme, our aim is to *identify* malicious nodes which generate polluted packets. After detecting such malicious attackers, one can simply block their communications to prevent further resource depletion in WMNs, e.g., blacklist them from the neighbor list, so as to defend against the pollution attack.

3 Detection Methodology

In this section, we propose our detection algorithms which are based on batch verification to identify pollution attackers, as well as the analytical methodology to quantify the performance measures of the algorithms.

Since we focus on backbones of WMNs which use network-coding enabled opportunistic routing protocol (e.g., *MORE*), we first give a brief overview of the MORE protocol. In MORE, the source node sends packets in generations, and each generation contains n native packets. When the source node is permitted to transmit, it will broadcast coded packets which are the linear combination of the native packets instead of directly broadcasting the native packets. A MORE header is attached to each coded packet which contains a list of potential forwarders. The source node chooses all its downstream nodes which have a lower ETX [7] distance to the destination as the potential forwarders. For a forwarder, when it receives a packet, it checks whether it is in the "forwarder list" or not, and also checks whether the packet is innovative or not. If yes, it makes a number of transmissions wherein each transmitted packet is also a linear combination of all its received packets in the same generation. For the destination node, if it receives n independent packets, it sends an acknowledgment to inform the source to transmit next generation.

Specifically, consider the example in Figure 1 where the source (node 1) sends coded packets CP_1, CP_2 and CP_3 to the destination (node 5) at time t_1, t_2 and t_3 respectively. When node 1 broadcasts these packets, both node 2 and node 3 can overhear these packet transmissions. Since node 3 is closer to the destination than the source on the ETX metric and these packets are innovative, it participates in the forwarding, e.g., after receiving CP_3, node 3 will perform multiple transmissions, and in each transmission, all received packets are encoded, e.g., it broadcasts $CP_1 \oplus CP_2 \oplus CP_3$.

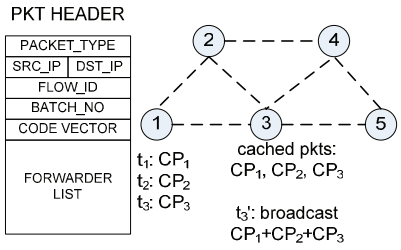


Fig. 1: Any two nodes that are connected with a dash line can communicate with each other. ETX value of each link is 1.

Before we present our detection algorithms and the theoretic analysis, we first list all notations that are used in this paper, and the formal definitions will be presented later.

A	: the detector node we focus on.
B	: a neighbor of the detector node A.
$\mathcal{N}_A$	: the set of neighbors of the detector node A.
N	: number of neighbors, i.e., $N = \lvert\mathcal{N}_A\rvert$.
K	: number of malicious neighbors or attackers.
$\mathcal{F}(t)$	: forwarder set at round t.
$\mathcal{S}(t)$	: suspicious set at round t.
α	: forwarding probability.
δ	: imitation probability.
p	: probability of ignoring a round.
$\mathcal{D}$	: number of detectable rounds.
$\mathcal{R}$	: total number of detection rounds.
$\mathcal{P}_{fn}(t)$	: probability of false negative at round t.
$\mathcal{P}_{fp}(t)$	: probability of false positive at round t.
p_s	: selection probability.
$\mathcal{F}^0(t)$	: forwarder set at round t with no attacker.
$\mathcal{F}^{\geq 1}(t)$	: forwarder set at round t with at least one attacker.
$\mathcal{P}_c$	: probability of correct detection.
$\mathcal{P}_w$	: probability of wrong detection.
$\mathcal{P}_m$	: probability of miss detection.

3.1 Core Idea of the Detection Algorithms

Since the detection algorithms we propose are *fully distributed*, i.e., each legitimate node in a WMN can execute the detection algorithms in an asynchronous fashion, in this paper, we only focus on a particular legitimate node, say node A, and describe its operations to identify its malicious neighbors. Let $\mathcal{N}_A$ be the set representing the neighbors of node A and we assume $|\mathcal{N}_A| = N$. Among these N neighbors, we assume that K ($K \geq 1$) of them are malicious attackers.

Let us first look at the behaviors of a malicious node, when it is ready to broadcast a packet, it may choose one of the following actions:

1) with probability δ, *imitates* as a legitimate node by performing correct coding operation and broadcasting a valid encoded packet, or
2) with probability $(1-\delta)$, *broadcasts a polluted packet*.

Here δ is called *the imitation probability*. The reason why malicious nodes may imitate legitimate nodes is to thwart the detection so as to reduce the chance of being detected.

On the other hand, for any legitimate node, it strictly follows the routing protocol. Specifically, a legitimate node maintains two buffers, *verified_buffer* and *unverified_buffer*. Every time when it is going to forward packets, it only encodes the packets in the verified_buffer. On receiving a new packet, it buffers the packet into the unverified_buffer. When a checksum packet arrives, it verifies those packets in the unverified_buffer based on the time based checksum verification scheme. If the batch verification matches, then all verified packets are shifted to verified_buffer, otherwise, all packets are discarded. Note that, by dropping the packets when batch verification does not match, epidemic spreading of polluted packets is avoided so that all packets forwarded by legitimate nodes are valid.

We define the duration of detector A receiving packets (including one valid checksum packet) and performing batch verification as a *round*. In other words, round t is the time period from right after the $(t-1)^{th}$ verification to right after the t^{th} verification performed by node A. At round t, the t^{th} batch verification is performed. If the verification does not match, then there must be some polluted packets and node A can not determine which packets are polluted, so it can only discard all of them to avoid further epidemic spreading. One thing we want to emphasize is that, to achieve high throughput, one can not verify the packets in the unverified_buffer one by one when receives a checksum packet but perform batch verification instead. At round t, some neighbors of detector A may forward packets to it and others may not. We define $\mathcal{F}(t)$ as the set of neighbors of node A which forward innovative packets to it at round t, and we call it *the forwarder set*. On the other hand, $\bar{\mathcal{F}}(t)$ is denoted as the set of neighbors which do not forward at round t. We have $\mathcal{N}_A = \mathcal{F}(t) \cup \bar{\mathcal{F}}(t)$.

The core idea of our algorithms is that, at round t, node A determines the suspicious set $\mathcal{S}(t)$, which contains all of its *potentially malicious neighbors* until the end of round t. As time proceeds in rounds, node A can *shrink* the suspicious set $\mathcal{S}(t)$ so that eventually, it only contains malicious neighbors of node A. After the detection, node A claims that a node is an attacker if and only if it stays in the suspicious set.

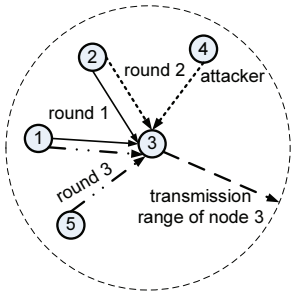


Fig. 2: Illustration of our detection mechanism.

Let us use a simple example to illustrate the core idea. Consider the example in Figure 2 wherein node 3 has four neighbors and node 4 is a malicious node. We initialize the suspicious set at round 0 as $\mathcal{S}(0) = \mathcal{N}_3 = \{1, 2, 4, 5\}$. We assume that in the first round, node 1 and node 2 forward packets to node 3, i.e., $\mathcal{F}(1) = \{1, 2\}$, and the batch verification matches, then node 3 knows that node 1 and node 2 must be legitimate nodes, so the suspicious set shrinks as $\mathcal{S}(1) = \mathcal{S}(0) \cap \bar{\mathcal{F}}(1) = \{4, 5\}$. In the second round, if node 2 and node 4 forward packets, since node 4 is a malicious node and it forwards polluted packets, the batch verification does not match. Node 3 knows that there is at least one attacker in the forwarder set $\mathcal{F}(2) = \{2, 4\}$, but it can not be sure whether all attackers are in the forwarder set or not as it does not know how many attackers exist, so $\mathcal{S}(1)$ keeps unchanged. However, if node 1 and node 5 forward in the third round and the batch verification matches, then node 3 can shrink the suspicious set via $\mathcal{S}(3) = \mathcal{S}(2) \cap \bar{\mathcal{F}}(3) = \{4\}$. Therefore, by shrinking the suspicious set, node 3 detects the attacker, node 4.

Besides developing algorithms to implement the above idea, we also quantify the performance measures of the algorithms. In particular, the performance measures include:

- $\mathcal{P}_{fn}(t)$, probability of false negative,
- $\mathcal{P}_{fp}(t)$, probability of false positive, and
- $E[\mathcal{R}]$, expected number of rounds needed for detection until the suspicious set only contains malicious nodes.

The first two performance measures quantify the *accuracy* of our detection algorithms, while the last performance measure quantifies the *efficiency*, in terms of time complexity, of our detection procedure. Precisely, $\mathcal{P}_{fn}(t)$ is defined as the probability of a malicious node being wrongly removed from the suspicious set $\mathcal{S}(t)$ at the end of round t. Since we claim that a node is an attacker if and only if it belongs to $\mathcal{S}(t)$ after t rounds, $\mathcal{P}_{fn}(t)$ is in fact the *probability of false negative*. On the other hand, $\mathcal{P}_{fp}(t)$ is defined as the probability of a randomly chosen node in $\mathcal{S}(t)$ being a legitimate node, which is in fact the *probability of false positive*. Lastly, r.v. $\mathcal{R}$ is used to denote the number of rounds needed for detection until all nodes in $\mathcal{S}(t)$ are malicious nodes or $\mathcal{S}(t)$ is empty.

In the following, we separate the analysis into two cases to illustrate our detection algorithms:

1) malicious nodes do not imitate the action of legitimate nodes, or the imitation probability $\delta = 0$;
2) malicious nodes may imitate the action of legitimate nodes to reduce the chance of being detected, or $\delta > 0$.

3.2 Attackers with Imitation Probability $\delta = 0$

Consider the case when $\delta = 0$, i.e., when a malicious node attempts to transmit, it always broadcasts polluted packets. As we stated before, since our detection algorithm is fully distributed, we only focus on a particular detector, node A. We initialize $\mathcal{S}(0) = \mathcal{N}_A$. As time proceeds in rounds, nodes in $\mathcal{N}_A$ are classified into different types according to their actions. Based on this classification, the suspicious set $\mathcal{S}(t)$ which contains the potential malicious nodes in $\mathcal{N}_A$ shrinks. Eventually, the suspicious set $\mathcal{S}(t)$ only contains malicious nodes, then node A can claim that it has identified the attackers. Note that, based on the definition of probability of false positive, if it is zero, then it means that all nodes in the suspicious set are attackers. Therefore, the stopping criteria of the detection algorithm can be designed based on probability

of false positive. Specifically, one can stop shrinking and take all nodes in $\mathcal{S}(t)$ as attackers when probability of false positive is smaller than some predefined threshold θ. The detection algorithm can be described as follows.

Algorithm 1 Detection Algorithm When $\delta = 0$

1: $\mathcal{S}(0) = \mathcal{N}_A$;
2: **repeat**
3: **if** the batch verification matches **then**
4: $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1) \cap \bar{\mathcal{F}}(t)$;
5: **else**
6: $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1)$;
7: **end if**
8: **until** $\mathcal{P}_{fp}(t) < \theta$

The rationale of Algorithm 1 is as follows. In each round, detector A performs the batch verification, if the verification matches, then it means that the malicious nodes cannot be in the set of nodes which performed forwarding at that round. Since we assume the malicious nodes always forward polluted packets ($\delta = 0$), if the malicious nodes forward, then the verification must not match. On the other hand, if there is a mismatch of the batch verification, then at least one malicious node forwarded at that round. However, because of the possibility of existing multiple malicious nodes, maybe only some malicious nodes forwarded, i.e., they belong to the forwarder set, but others did not. Therefore, we cannot distinguish the malicious nodes from the legitimate nodes and the suspicious set remains unchanged.

Now, let us look at what information node A can obtain at each round. Note that, at round t, node A knows the forwarder set $\mathcal{F}(t)$ which contains the neighbors that forward packets to it. Moreover, when node A performs batch verification, it knows whether the verification matches or not. We call a round where the suspicious set shrinks *the detectable round* and use a 0-1 random variable $d(t)$ to indicate it. Based on Algorithm 1, $d(t)$ equals to one if the batch verification matches at round t, and zero otherwise. We call $d(t)$ the detectable round indicator. Node A maintains the forwarder set $\mathcal{F}(t)$ and the detectable round indicator $d(t)$ which are obtained at each round. We use the tuple of these two parameters to define the state of node A at round t, which is denoted as $s(t)$. Therefore, we have $s(t) = (\mathcal{F}(t), d(t))$. The collection of all these states composes the detection history of node A and we denote it as $\mathcal{H}(t)$, i.e., $\mathcal{H}(t) = (\mathcal{F}(1), d(1)), (\mathcal{F}(2), d(2)), ..., (\mathcal{F}(t), d(t))$.

To quantify the performance of the algorithm, we first look at how the detection history $\mathcal{H}(t)$ is generated. Firstly, we assume that the nodes' forwarding decisions are independent and identically distributed. And at each round, a node forwards packets with probability α which is called *the forwarding probability*. In fact, this independent and identical assumption is reasonable since in a wireless mesh network, fairness is a built-in feature in the medium access control (MAC) protocol, e.g., in 802.11, a node cannot monopolize the wireless resource by repeatedly sending packets. When the communication channel is free, all nodes will compete for the channel. In other words, nodes have the same chance to participate in forwarding. Another thing is that due to the property of the MORE routing protocol, specifically, if a received packet is not innovative, it will be discarded immediately, and a generation only contains 32 independent packets in the common setting under MORE, moreover, a node may receive multiple checksum packets during the period of transmitting one generation, the forwarding probability α is less than one for most cases.

To derive the forwarding probability at each round, we use a random variable $I_B(\tau)$ to indicate whether neighbor B forwards at round τ or not, i.e., if B forwards, then $I_B(\tau) = 1$, otherwise, $I_B(\tau) = 0$. Based on our assumption, we have $I_B(\tau)$s are independent and identically distributed for all $B \in \mathcal{N}_A$. Now, the forwarding probability can be can be represented by $\alpha(\tau) = Prob\{I_B(\tau) = 1\}$. Note that, at round τ, the number of nodes which forward is $|\mathcal{F}(\tau)|$, we have

$$E[\sum\nolimits_{B\in\mathcal{N}_A} I_B] = N \cdot Prob\{I_B(\tau) = 1\} = N\alpha(\tau) = |\mathcal{F}(\tau)|.$$

Therefore, the forwarding probability is $\alpha(\tau) = \frac{|\mathcal{F}(\tau)|}{N}$.

Now, let us derive the performance measures of Algorithm 1. We assume that the number of malicious neighbors of node A is K ($K \geq 1$). Based on Algorithm 1, we can see that no malicious node will be removed from the suspicious set $\mathcal{S}(t)$. Therefore, the probability of false negative is simply zero.

$$\mathcal{P}_{fn}(t) = 0. \tag{3}$$

To derive the probability of false positive, which is the probability that a randomly chosen node in $\mathcal{S}(t)$ is a legitimate node, we first compute the probability of a node being removed from the suspicious set at round τ. We first consider a malicious node, and use $P_M(\tau)$ to denote this probability. Since $\delta = 0$, $P_M(\tau)$ is just 0. For a legitimate node, we use $P_L(\tau)$ to denote the probability. Observe that, a legitimate node will be removed from $\mathcal{S}(t)$ only when it forwards valid packets in some detectable round τ. So $P_L(\tau)$ equals to $\alpha(\tau)$. Therefore, the probability of false positive is:

$$\begin{aligned}
\mathcal{P}_{fp}(t) &= P(B \text{ is legitimate} \mid \mathcal{H}(t) \& B \in \mathcal{S}(t)) \\
&= \frac{P(B \text{ is legitimate } \& B \in \mathcal{S}(t) \mid \mathcal{H}(t))}{P(B \in \mathcal{S}(t) \mid \mathcal{H}(t))} \\
&= \frac{\frac{N-K}{N}\prod_{\tau=1,d(\tau)=1}^{t}(1-P_L(\tau))}{\frac{N-K}{N}\prod\limits_{\substack{\tau=1\ldots t\\ d(\tau)=1}}(1-P_L(\tau)) + \frac{K}{N}\prod\limits_{\substack{\tau=1\ldots t\\ d(\tau)=1}}(1-P_M(\tau))} \\
&= \frac{(N-K)\prod_{\tau=1,d(\tau)=1}^{t}(1-\alpha(\tau))}{(N-K)\prod_{\tau=1,d(\tau)=1}^{t}(1-\alpha(\tau)) + K}.
\end{aligned} \tag{4}$$

Note that K is the number of attackers in the neighborhoods, and node A is unaware of its value. However, by running the algorithm for enough number of rounds, the suspicious set will only contain malicious attackers. In other words, the size of the suspicious set converges to K, i.e., $|\mathcal{S}(t)| \to K$. Therefore, one can use $|\mathcal{S}(t)|$ to approximate K when compute probability of false positive, and this approximation converges to the accurate value. In other words, it is a good choice to use this approximation to design the stopping criteria. Based

on this approximation, $\mathcal{P}_{fp}(t)$ is computed as follows.

$$\mathcal{P}_{fp}(t) = \frac{(N-|\mathcal{S}(t)|)\prod_{\tau=1,d(\tau)=1}^{t}(1-\alpha(\tau))}{(N-|\mathcal{S}(t)|)\prod_{\tau=1,d(\tau)=1}^{t}(1-\alpha(\tau))+|\mathcal{S}(t)|}. \quad (5)$$

To derive the third performance measure $E[\mathcal{R}]$, we assume that the forwarding probability is the same at every round and use α to denote it. Since the suspicious set cannot shrink in every round, we define those rounds where the suspicious set shrinks as *detectable rounds* and use random variable $\mathcal{D}$ to denote the number of detectable rounds that node A needs to detect malicious nodes. To compute the distribution of $\mathcal{D}$, observe that, a legitimate node is removed from the suspicious set only when it forwards at a detectable round, which happens with probability α. Therefore,

$$\begin{aligned} P(\mathcal{D} \le d) &= P(\text{after } d \text{ detectable rounds, nodes in the suspicious node set are all malicious}) \\ &= P(\text{for each legitimate neighbor, it forwards in at least one detectable round}) \\ &= \left(1-(1-\alpha)^d\right)^{N-K}. \end{aligned} \quad (6)$$

For Algorithm 1, a detectable round happens when no polluted packet is forwarded to node A, the corresponding probability is $p_d = (1-\alpha)^K$. Given the number of detectable rounds $\mathcal{D}$, the conditional distribution $P(\mathcal{R}=r|\mathcal{D}=d)$ is again a negative binomial distribution, or,

$$\begin{aligned} P(\mathcal{R}=r) &= \sum_{d=1}^{r} P(\mathcal{D}=d)P(\mathcal{R}=r|\mathcal{D}=d) \quad (7) \\ &= \sum_{d=1}^{r} \binom{r-1}{d-1}(p_d)^d(1-p_d)^{r-d}P(\mathcal{D}=d). \end{aligned}$$

With this probability distribution, $E[\mathcal{R}]$ can be easily computed via $E[\mathcal{R}] = \sum_{r=1}^{\infty} rP(\mathcal{R}=r)$.

3.3 Attackers with Imitation Probability $\delta > 0$

Let us consider a more interesting case where a malicious node can imitate as a legitimate node by forwarding a valid packet with probability δ ($\delta > 0$). In other words, when a malicious node is prepared to broadcast a packet, it does not modify the packet but performs correct network coding with probability δ. Under this situation, if the verification does not match, node A knows it must have received some polluted packets. However, due to the existence of multiple attackers, node A can not be certain whether all malicious nodes are in the forwarder set or not. On the other hand, if the verification matches, node A still faces with the problem of accurately detecting the malicious nodes since they may pretend to be legitimate nodes. As discussed before, the goal of malicious nodes is to reduce the system performance or damage the system by injecting polluted packets. However, the imitation actions violate this objective. Moreover, the damage that the attackers can cause in one round is limited as they can only make the packets forwarded by legitimate nodes in that round be discarded. Therefore, the imitation probability δ cannot be too large for rational attackers. Based on this fact, we propose a "*randomized detection algorithm*" as follows.

Algorithm 2 Detection Algorithm When $\delta > 0$

```
repeat
    if the batch verification matches then
        with probability 1 − p:   S(t) ← S(t − 1) ∩ F̄(t);
    else
        S(t) ← S(t − 1);
    end if
until P_fp(t) < θ
```

Let us derive the performance of Algorithm 2. Firstly, note that, a round is detectable only when the batch verification matches and it is not ignored. Secondly, if a malicious node pretends to be a legitimate node to forward valid packets in some detectable round, then it will be removed from the suspicious set, i.e., it evades the detection. This error is characterized by probability of false negative, $\mathcal{P}_{fn}(t)$. Formally, this probability is derived as follows.

$$\begin{aligned} \mathcal{P}_{fn}(t) &= P(\text{after } t \text{ rounds, the malicious node is not in the suspicious set } \mathcal{S}(t) \mid \mathcal{H}(t)) \\ &= P(\text{the malicious node forwards valid packets in at least one detectable round} \mid \mathcal{H}(t)) \\ &= 1 - \prod_{\tau=1,d(\tau)=1}^{t} \frac{1-\alpha(\tau)}{1-\alpha(\tau)+\alpha(\tau)\delta}, \end{aligned} \quad (8)$$

where $\alpha(\tau) = \frac{|\mathcal{F}(\tau)|}{N}$ is the forwarding probability at round τ, and $d(\tau) = 1$ only when round τ is a detectable round.

To derive the probability of false positive, note that, the probability that a malicious node is removed from the suspicious set at some detectable round τ, i.e., $P_M(\tau)$, is $\frac{\alpha(\tau)\delta}{1-\alpha(\tau)+\alpha(\tau)\delta}$. Similarly, for a legitimate node, the corresponding probability is $\alpha(\tau)$, i.e., $P_L(\tau) = \alpha(\tau)$. Based on Equation (4), the probability of false positive for Algorithm 2 is

$$\mathcal{P}_{fp}(t) = \frac{(N-K)\prod_{\tau=1,d(\tau)=1}^{t}(1-P_L(\tau))}{(N-K)\prod\limits_{\substack{\tau=1\ldots t\\ d(\tau)=1}}(1-P_L(\tau))+K\prod\limits_{\substack{\tau=1\ldots t\\ d(\tau)=1}}(1-P_M(\tau))}.$$

Again, we have to approximate K. Since the probability of each malicious node being in the suspicious set is $1-\mathcal{P}_{fn}(t)$, K can be approximated as $\frac{|\mathcal{S}(t)|}{1-\mathcal{P}_{fn}(t)}$. Therefore, $\mathcal{P}_{fp}(t)$ is

$$\mathcal{P}_{fp}(t) = \frac{N-\frac{|\mathcal{S}(t)|}{1-\mathcal{P}_{fn}(t)}}{(N-\frac{|\mathcal{S}(t)|}{1-\mathcal{P}_{fn}(t)})+\frac{|\mathcal{S}(t)|}{1-\mathcal{P}_{fn}(t)}\prod\limits_{\substack{\tau=1\ldots t\\ d(\tau)=1}}\frac{1-P_M(\tau)}{1-P_L(\tau)}}. \quad (9)$$

To derive $E[\mathcal{R}]$, we first derive the distribution of number of detectable rounds $\mathcal{D}$, based on Equation (6), we have

$$P(\mathcal{D} \le d) = \left(1-(1-\alpha)^d\right)^{N-K}. \quad (10)$$

Observe that, a detectable round only happens when the verification matches and the round is chosen for detection. The corresponding probability is $p_d = (1-\alpha+\alpha\delta)^K(1-p)$. Given the number of detectable rounds $\mathcal{D}$, the conditional distribution

$P(\mathcal{R}=r|\mathcal{D}=d)$ is a negative binomial distribution, or,

$$P(\mathcal{R}=r)=\sum_{d=1}^{r} P(\mathcal{D}=d)P(\mathcal{R}=r|\mathcal{D}=d)$$
$$=\sum_{d=1}^{r}\binom{r-1}{d-1}(p_d)^d(1-p_d)^{r-d}P(\mathcal{D}=d), \quad (11)$$

where $p_d=(1-\alpha+\alpha\delta)^K(1-p)$ and $P(\mathcal{D}=d)$ can be easily derived from Equation (10). Given this distribution, $E[\mathcal{R}]$ can be easily computed by $E[\mathcal{R}]=\sum_{r=1}^{\infty} rP(\mathcal{R}=r)$.

3.4 Improvement on $\mathcal{P}_{fn}$

In the last subsection, we address the case where malicious nodes may pretend to be legitimate nodes to forward valid packets. When the algorithm runs for sufficient number of rounds, we can guarantee that all nodes in the suspicious set are malicious nodes, i.e., probability of false positive converges to 0. However, since the malicious node may evade the detection, probability of false negative is not 0. To detect all malicious attackers, observe that, when Algorithm 2 runs for sufficient number of rounds, all nodes in the suspicious set $\mathcal{S}(t)$ are malicious. The insufficiency is that only a fraction of malicious nodes stay in $\mathcal{S}(t)$. Intuitively, one can remove those detected malicious nodes (i.e., blacklist them for further data exchange), then repeat the detection process again. After repeating the detection process multiple times, one can be certain in removing all malicious nodes from the neighbor list. The enhanced algorithm is as follows.

Algorithm 3 Enhanced Detection Algorithm

```
loop
  S(0) ← N_A;
  repeat
    if the batch verification matches then
      with probability 1−p:  S(t) ← S(t−1) ∩ F̄(t);
    else
      S(t) ← S(t−1);
    end if
  until P_fp(t) < θ
  remove nodes in S(t) from neighbors: N_A ← N_A \ S(t);
end loop
```

One important performance measure for this enhanced algorithm is that how many times we have to repeat until all attackers are identified. Note that, for each execution of Algorithm 2, a fraction of $1-\mathcal{P}_{fn}(t)$ attackers are identified, so as long as $(\mathcal{P}_{fn}(t))^\beta$ is small enough, one can claim that all attackers are identified. Therefore, one can roughly estimate the total number of detection rounds for identifying all attackers as $\beta E[\mathcal{R}]$ where $E[\mathcal{R}]$ is derived by Equation (11). In our simulation, the results show that one can identify all attackers in very small number of executions.

4 Generalized Detection Algorithm

In the last section, we present the detection algorithms and the analytic methodology to quantify their performance measures. The core idea of our algorithms is to shrink the suspicious set until it only contains malicious nodes. Therefore, making sure that the suspicious set can get shrunk is the key point to guarantee the effectiveness of our algorithms. However, based on our detection algorithms (Algorithm 1 and Algorithm 2), the suspicious set cannot shrink if the batch verification does not match. Therefore, if the detector receives polluted packets in most rounds, which may happen when multiple attackers exist and they cooperate with each other to launch powerful attack, then the number of rounds needed for detection may become very large and the algorithm may even fail to detect malicious attackers. We call this case the *cooperative pollution attack* to distinguish from the *ordinary pollution attack* we have discussed in last section. In this section, we provide a general randomized detection algorithm which cannot only be used to speed up the detection when ordinary pollution attack is performed, but can also be used to identify the malicious attackers when cooperative attack is launched.

4.1 Detection Algorithm

The basis of the generalized algorithm is as follows. Firstly, to defend against the cooperative attack, every time when the detector receives a checksum packet and performs batch verification, it first randomly selects every forwarder with probability p_s which is called the *selection probability*, and only encodes the packets received from these forwarders for batch verification. We still use $\mathcal{F}(t)$ to denote the set of forwarders which are selected for batch verification at round t. Secondly, at round t, if the batch verification does not match, even if we cannot shrink the suspicious set because maybe only a part of malicious nodes are in the forwarder set $\mathcal{F}(t)$, but we can make sure that at least one attacker exists in the forwarder set $\mathcal{F}(t)$ as the polluted packets can not be forwarded by legitimate nodes. To speed up the detection, we also make use of the information gained from the rounds where batch verification does not match. We first list two notations.

- $\mathcal{F}^0(t)$: the set of forwarders which are selected for batch verification at round t, and the batch verification matches, i.e., no malicious node exists in this set without considering the imitation behaviors.
- $\mathcal{F}^{\geq 1}(t)$: the set of forwarders which are selected for batch verification at round t, and the batch verification does not match, i.e., *at least* one of them is malicious.

We call $\mathcal{F}^0(t)$ the *type one forwarder set* and $\mathcal{F}^{\geq 1}(t)$ the *type two forwarder set*. We define $\mathcal{S}(t)$ as the suspicious set which stores all the detection information until round t. We initialize it as an empty set, i.e., $\mathcal{S}(0)=\emptyset$. The procedure for shrinking the suspicious set at round t is presented in Algorithm 4.

The rationale of the generalized algorithm can be illustrated as follows. By randomly selecting forwarders for batch verification, one can prevent the attackers cooperatively injecting polluted packets which may make the batch verification do not match at every round. Moreover, randomness also brings benefits in terms of distinguishing the forwarders. After the batch verification at round t, if the verification does not match, then we can make sure that at least one attacker exists in the forwarder set, i.e., we get a type two forwarder set $\mathcal{F}^{\geq 1}(t)$.

Algorithm 4 Generalized Detection Algorithm at Round t

1: $\mathcal{F}(t) \leftarrow \emptyset$;
2: **for** every forwarder B at round t **do**
3: with probability p_s: $\mathcal{F}(t) \leftarrow \mathcal{F}(t) \cup \{B\}$;
4: **end for**
5: encode the packets received from forwarders in $\mathcal{F}(t)$ and perform batch verification;
6: **if** the batch verification matches **then**
7: $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1) \wedge \mathcal{F}^0(t)$;
8: **else**
9: $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1) \wedge \mathcal{F}^{\geq 1}(t)$;
10: **end if**

On the other hand, if the batch verification matches, we get a type one forwarder set $\mathcal{F}^0(t)$. Although the forwarder set may contain attackers as they may pretend as good nodes to forward correct packets, we still utilize this forwarder set for detection. The reason is that as we stated in Section 3.3, the imitation probability δ cannot be too large. Moreover, the nodes in $\mathcal{F}^0(t)$ are randomly selected from forwarders with probability p_s. Therefore, one can still avoid high detection error even if utilize the type one forwarder sets.

To derive $\mathcal{S}(t)$, we have to define the operations on $\wedge$ since we have to compute $\mathcal{S}(t-1) \wedge \mathcal{F}^s(t)$ where s is either "0" or "≥ 1". We have the following cases.

Case 1: $\mathcal{S}(t-1) = \emptyset$. Since $\mathcal{S}(t)$ is initialized as an empty set, this case happens when the algorithm starts. The operation under this case is defined as follows.

$$\mathcal{S}(t-1) \wedge \mathcal{F}^s(t) = \begin{cases} \emptyset, & \text{if } s \text{ is "0"}, \\ \{\mathcal{F}^s(t)\}, & \text{if } s \text{ is "}\geq 1\text{"}. \end{cases} \quad (12)$$

The reason why we ignore the type one forwarder set is that we are only uncertain about the types of nodes that are in type two forwarder sets, so we only have to keep the type two forwarder sets in the suspicious set.

Case 2: $\mathcal{S}(t-1) \neq \emptyset$ and $\mathcal{F}^s(t)$ is a type one forwarder set, i.e., s is "0", or $\mathcal{F}^s(t)$ is a type two forwarder set but there exists at least one forwarder set $\mathcal{F}^s(\tau)$ in the suspicious set such that either $\mathcal{F}^s(\tau)$ is a subset of $\mathcal{F}^s(t)$ or $\mathcal{F}^s(t)$ is a subset of $\mathcal{F}^s(\tau)$ holds. The operation under this case is

$$\mathcal{S}(t-1) \wedge \mathcal{F}^s(t) = \{\mathcal{F}^s(\tau) \wedge \mathcal{F}^s(t) | \mathcal{F}^s(\tau) \in \mathcal{S}(t-1)\}. \quad (13)$$

Now we define the operation of $\wedge$ between the two forwarder sets $\mathcal{F}^s(\tau)$ and $\mathcal{F}^s(t)$ under case 2. We first consider s is "0".

$$\mathcal{F}^{\geq 1}(\tau) \wedge \mathcal{F}^0(t) = \mathcal{F}^{\geq 1}(\tau) \backslash \mathcal{F}^0(t). \quad (14)$$

The rationale of this operation is that since the type one forwarder set $\mathcal{F}^0(t)$ does not contain malicious node without considering the imitation behavior, one can simply remove them from the type two forwarder set. However, since malicious node may pretend to be a good node, this operation may cause some detection error. Specifically, maybe the reduced set does not contain any malicious node but we still believe that it contains at least one, and we will quantify the error later.

If s is "≥ 1" under case 2, the operation is defined as

$$\mathcal{F}^{\geq 1}(\tau) \wedge \mathcal{F}^{\geq 1}(t) = \begin{cases} \mathcal{F}^{\geq 1}(\tau), & \text{if } \mathcal{F}^{\geq 1}(\tau) \subseteq \mathcal{F}^{\geq 1}(t), \\ \mathcal{F}^{\geq 1}(t), & \text{if } \mathcal{F}^{\geq 1}(t) \subseteq \mathcal{F}^{\geq 1}(\tau). \end{cases} \quad (15)$$

The rationale of this operation is that if a malicious node exists in some set, then it must be in its superset, so we do not need to keep the redundant information in the suspicious set.

Case 3: If neither case 1 nor case 2 hold, we simply add the forwarder set to the suspicious set. Note that, in this case, the forwarder set $\mathcal{F}^s(t)$ must be of type two, i.e., s is "≥ 1", and it can not be a subset or superset of any element in the suspicious set. In other words, it is just like that this forwarder set contains some *independent* information, so we have to keep it. Mathematically, we have

$$\mathcal{S}(t-1) \wedge \mathcal{F}^s(t) = \mathcal{S}(t-1) \cup \{\mathcal{F}^s(t)\}. \quad (16)$$

By now, we have defined all operations on $\wedge$, i.e., Equation (12)-(16). Note that, based on our definitions, the suspicious set will only contain type two forwarder set, so we do not need the operation of $\wedge$ between two type one forwarder sets. In the following, we will provide the rule of making decisions on the types of nodes. In other words, how to determine whether a node is a malicious node or not. Before we do that, we first show that every forwarder set in the suspicious set can shrink to a singleton set or an empty set and the result is stated in the following theorem.

Theorem *1:* For every forwarder set in the suspicious set, by using the operations defined in Equation (12)-(16), it can finally shrink to one of the following three sets.

- case 1: $\{B_m\}^{\geq 1}$ where B_m is a malicious node;
- case 2: $\{B_g\}^{\geq 1}$ where B_g is a good node;
- case 3: $\emptyset$.

Proof: For every forwarder set $\mathcal{F}^s(\tau)$ in the suspicious set, it must be a type two forwarder set, i.e., s is "≥ 1". Let $|\mathcal{F}(\tau)| = n$. Without loss of generality, we assume that k of the n nodes in the forwarder set are attackers, we have $1 \leq k \leq \min\{K, n\}$. At round t, the forwarder set can shrink in two cases. Firstly, if the batch verification matches, then the nodes in $\mathcal{F}^0(t)$ will be removed. So the probability that the size of the forwarder set $\mathcal{F}^{\geq 1}(\tau)$ decreases at least by one is

$$p_1 = (1-\alpha(1-\delta)p_s)^{K-k}[(1-\alpha(1-\delta)p_s)^k - (1-\alpha p_s)^n]. \quad (17)$$

The second case that $\mathcal{F}^{\geq 1}(\tau)$ can shrink is that the forwarder set at round t is of type two and it is a subset of $\mathcal{F}^{\geq 1}(\tau)$, i.e. $\mathcal{F}^{\geq 1}(t) \subset \mathcal{F}^{\geq 1}(\tau)$. The probability that the size of the forwarder set $\mathcal{F}^{\geq 1}(\tau)$ decreases at least by one in this case is

$$p_2 = (1-\alpha p_s)^{N-n}[1-(1-\alpha(1-\delta)p_s)^k - (\alpha p_s)^n]. \quad (18)$$

Note that both p_1 and p_2 is positive, so the probability that the forwarder set shrinks at every round, $p_1 + p_2$, is positive. Therefore, after enough number of rounds, it is sure to shrink the forward set $\mathcal{F}^{\geq 1}(\tau)$ to a singleton set or an empty set. ∎

Now we present the decision rule which is used to identify attackers. Since every forwarder set in the suspicious set can shrink to a singleton set or an empty set, we will take the node in the singleton set as attacker and remove it from neighborhoods. Note that, we also have to remove all forwarder sets which contain the identified attacker from the suspicious set. Formally, the decision rule can be described as follows.

Decision rule: take B as an attacker iff $\{B\}^{\geq 1} \in \mathcal{S}(t)$. (19)

Note that, based on Theorem 1, the node in the singleton set may be a good node, so detection error may happen when use the decision rule presented above. We will quantify the error later. One thing we want to emphasize is that, one can plug our algorithm into the routing protocol. In other words, the algorithm will keep running in every detector node as long as there is packet forwarding through the node. Specifically, every detector node maintains one suspicious set which stores the type two forwarder sets according to our algorithm and the operations we defined before. As long as a forwarder set shrinks to a singleton set, take the node as attacker and remove it from the neighborhoods. Note that, if no attacker exits in the neighborhoods, then the suspicious set will keep being an empty set as no type two forwarder set appears.

4.2 Performance Evaluation

Now, we quantify the performance of the generalized detection algorithm. Based on Theorem 1, one type two forwarder set can shrink to be $\{B_m\}^{\geq 1}$, $\{B_g\}^{\geq 1}$ or $\emptyset$. Based on the decision rule, if $\{B_m\}^{\geq 1}$ happens, then one attacker is correctly identified and we call this case *correct detection*, if $\{B_g\}^{\geq 1}$ happens, then we wrongly take a good node as an attacker and we call this case *wrong detection*, if the forwarder set shrinks to be an empty set, then we just waste some rounds and we call this case *miss detection*. To quantify the performance of our algorithm, we only need to focus on the shrinkage process of a particular forwarder set in the suspicious set, and we are interested in the following four measures.

- $\mathcal{R}$: average number of rounds needed to shrink a forwarder set to $\{B_m\}^{\geq 1}$, $\{B_g\}^{\geq 1}$ or $\emptyset$.
- $\mathcal{P}_c$: probability of correct detection, i.e., the probability that the forwarder set shrinks to $\{B_m\}^{\geq 1}$.
- $\mathcal{P}_w$: probability of wrong detection, i.e., the probability that the forwarder set shrinks to $\{B_g\}^{\geq 1}$.
- $\mathcal{P}_m$: probability of miss detection, i.e., the probability that the forwarder set shrinks to $\emptyset$.

Intuitively, the first measure quantifies the efficiency of the algorithm and the last three measures quantify the effectiveness and we have $\mathcal{P}_c + \mathcal{P}_w + \mathcal{P}_m = 1$.

To derive the performance measures, we first focus on a particular forwarder set which contains n nodes and k of them are attackers when it is added into the suspicious set. For ease of presentation, we take the forwarder set as being in state (n, k). At each round, if i good nodes and j malicious nodes are removed from the forwarder set, then only n–i–j nodes left and $k-j$ of them are attackers, i.e., the forwarder set transits from state (n, k) to state $(n-i-j, k-j)$. We first compute such transition probabilities and the results are presented in the following lemma.

Lemma *1:* At each round, the probability that a forwarder set of state (n, k) transits to state $(n-i-j, k-j)$ $(0 \leq i \leq n-k, 0 \leq j \leq k)$ is computed as follows.

(1) if $i + j \neq 0$ and $j \neq k$, the transition probability is:

$$P_{ij}(n,k) = \left[1-\alpha(1-\delta)p_s\right]^{K-k}\left[\binom{k}{j}(1-\alpha p_s)^{k-j}(\alpha\delta p_s)^j\right]\left[\binom{n-k}{i}(1-\alpha p_s)^{n-k-i}(\alpha p_s)^i\right] + \left[1-\alpha p_s\right]^{N-n}\left[\binom{n-k}{i}(1-\alpha p_s)^i(\alpha p_s)^{n-k-i}\right]\left[\binom{k}{j}(1-\alpha p_s)^j((\alpha p_s)^{k-j} - (\alpha\delta p_s)^{k-j})\right]. \quad (20)$$

(2) if $i + j \neq 0$ and $j = k$, the transition probability is:

$$P_{ik}(n,k) = \left[\binom{n-k}{i}(1-\alpha p_s)^{n-k-i}(\alpha p_s)^i\right] \times (\alpha\delta p_s)^k\left[1-\alpha(1-\delta)p_s\right]^{K-k}. \quad (21)$$

(3) if $i = j = 0$, the transition probability is:

$$P_{00}(n,k) = 1 - \sum_{\substack{i=0 \\ i+j\neq 0}}^{n-k}\sum_{j=0}^{k} P_{ij}(n,k). \quad (22)$$

Now, we quantify the performance of shrinking a particular forwarder set which is at state (n, k) as follows.

Lemma *2:* For a particular forwarder set which is at state (n, k), the average number of rounds needed for shrinking it to $\{B_m\}^{\geq 1}$, $\{B_g\}^{\geq 1}$ or $\emptyset$ is

$$\mathcal{R}(n,k) = \frac{\sum_{\substack{i=0 \\ i+j\neq 0}}^{n-k}\sum_{j=0}^{k}(\mathcal{R}(n-i-j,k-j)+1)P_{ij}(n,k) + P_{00}(n,k)}{1-P_{00}(n,k)}, \quad (23)$$

where $\mathcal{R}(0,0) = 0$, $\mathcal{R}(1,0) = 0$ and $\mathcal{R}(1,1) = 0$.

Moreover, probability of correct detection, probability of wrong detection and probability of miss detection are

$$\mathcal{P}_x(n,k) = \frac{\sum_{\substack{i=0 \\ i+j\neq 0}}^{n-k}\sum_{j=0}^{k}\mathcal{P}_x(n-i-j,k-j)P_{ij}(n,k)}{1-P_{00}(n,k)}, \quad (24)$$

where x is c, w or m, $\mathcal{P}_c(0,0) = 0$, $\mathcal{P}_c(1,0) = 0$, $\mathcal{P}_c(1,1) = 1$, $\mathcal{P}_w(0,0) = 0$, $\mathcal{P}_w(1,0) = 1$, $\mathcal{P}_w(1,1) = 0$, $\mathcal{P}_m(0,0) = 1$, $\mathcal{P}_m(1,0) = 0$ and $\mathcal{P}_m(1,1) = 0$.

Proof: We use X_t to denote the state of the forwarder set at round t and let $T = \min\{t | X_t = (1,0), (1,1) \text{ or } (0,0)\}$. Then

$$\begin{aligned}
\mathcal{R}(n,k) &= E(T|X_0 = (n,k)),\\
\mathcal{P}_c(n,k) &= P\{X_T = (1,1)|X_0 = (n,k)\},\\
\mathcal{P}_w(n,k) &= P\{X_T = (1,0)|X_0 = (n,k)\},\\
\mathcal{P}_m(n,k) &= P\{X_T = (0,0)|X_0 = (n,k)\}.
\end{aligned}$$

By using the formula of total probability, we have

$$\begin{aligned}\mathcal{R}(n,k) =&\sum_{i=0}^{n-k}\sum_{j=0}^{k}E(T|X_1=(n-i-j,k-j),X_0=(n,k))\times\\ &P\{X_1=(n-i-j,k-j)|X_0=(n,k)\}\\ =&\sum_{i=0}^{n-k}\sum_{j=0}^{k}E(T|X_1=(n-i-j,k-j))P_{ij}(n,k)\\ =&\sum_{i=0}^{n-k}\sum_{j=0}^{k}\Big(\mathcal{R}(n-i-j,k-j)+1\Big)P_{ij}(n,k).\end{aligned}$$

By moving the $\mathcal{R}(n,k)$ in the right hand side to the left side, we have the result shown in Equation (23).

To derive the probability of correct detection, we have

$$\begin{aligned}\mathcal{P}_c(n,k) =&\sum_{i=0}^{n-k}\sum_{j=0}^{k}P\{X_T=(1,1)|X_1=(n-i-j,k-j)\}\times\\ &P\{X_1=(n-i-j,k-j)|X_0=(n,k)\}\\ =&\sum_{i=0}^{n-k}\sum_{j=0}^{k}P_c(n-i-j,k-j)P_{ij}(n,k).\end{aligned}$$

By moving the $\mathcal{P}_c(n,k)$ in the right hand side to the left side, we have the result shown in Equation (24). Similarly, $\mathcal{P}_w(n,k)$ and $\mathcal{P}_m(n,k)$ can also be derived. ∎

To release the condition that the forwarder set is at state (n,k) so as to derive the expectation of these performance measures, we only need to take into account all possible states in which a forwarder set can be when it is added into the suspicious set and take the average. Note that, the probability that a forwarder set is at state (n,k) is

$$\begin{aligned}P(n,k)=&\binom{N-K}{n-k}\Big(\alpha p_s\Big)^{n-k}\Big(1-\alpha p_s\Big)^{N-K-(n-k)}\times\\ &\binom{K}{k}\Big(1-\alpha p_s\Big)^{K-k}\Big((\alpha p_s)^k-(\alpha\delta p_s)^k\Big)/\Sigma,\end{aligned}\quad(25)$$

where

$$\begin{aligned}\Sigma=&\sum_{k=0}^{K}\sum_{n=k}^{N-(K-k)}\binom{N-K}{n-k}\Big(\alpha p_s\Big)^{n-k}\Big(1-\alpha p_s\Big)^{N-K-(n-k)}\times\\ &\binom{K}{k}\Big(1-\alpha p_s\Big)^{K-k}\Big((\alpha p_s)^k-(\alpha\delta p_s)^k\Big).\end{aligned}$$

Now, the expectation of the performance measures can be easily derived and the results are stated as follows.

Theorem *2:* For every forwarder set which is added into the suspicious set, the average number of rounds needed to shrink it to a singleton set or an empty set is

$$\mathcal{R}=\sum_{k=0}^{K}\sum_{n=k}^{N-(K-k)}\mathcal{R}(n,k)P(n,k).\quad(26)$$

where $\mathcal{R}(n,k)$ is derived in Equation (23) and $P(n,k)$ is derived in Equation (25).

Moreover, probability of correct detection, probability of wrong detection, and probability of miss detection are

$$\mathcal{P}_x=\sum_{k=0}^{K}\sum_{n=k}^{N-(K-k)}\mathcal{P}_x(n,k)P(n,k).\quad(27)$$

where x is c, w or m, $\mathcal{P}_x(n,k)$ is derived in Equation (24) and $P(n,k)$ is derived in Equation (25).

4.3 Detection Acceleration

In previous subsections, we present the generalized detection algorithm which is able to defend against the cooperative attack, and we also quantify the performance of the algorithm. In this subsection, we propose a new operation which is very useful to accelerate the shrinkage of the suspicious set.

Note that, after perform the operation defined in Equation (14), for a forwarder set $\mathcal{F}^{\geq 1}(\tau)$ in the suspicious set, maybe there exists another forwarder set $\mathcal{F}^{\geq 1}(\nu)$ which is a subset of $\mathcal{F}^{\geq 1}(\tau)$, i.e., $\mathcal{F}^{\geq 1}(\nu)\subset\mathcal{F}^{\geq 1}(\tau)$. If this happens, redundant information is kept in $\mathcal{F}^{\geq 1}(\tau)$. Intuitively, it can bring much benefit in terms of shrinking the forwarder set $\mathcal{F}^{\geq 1}(\tau)$ if we remove the redundant information. Formally, to speed up the detection, we perform the following operation on $\mathcal{S}(t)$ after round t ($t\geq 1$), and we call it *accelerating operation*.

$$\mathcal{S}(t)=\{\mathcal{F}^{\geq 1}(\tau)^*|\mathcal{F}^{\geq 1}(\tau)\in\mathcal{S}(t)\},\quad(28)$$

where $\mathcal{F}^{\geq 1}(\tau)^*$ is computed as follows.

$$\mathcal{F}^{\geq 1}(\tau)^*=\begin{cases}\mathcal{F}^{\geq 1}(\nu), & \text{if } \exists\mathcal{F}^{\geq 1}(\nu)\in\mathcal{S}(t)\&\mathcal{F}^{\geq 1}(\nu)\subset\mathcal{F}^{\geq 1}(\tau),\\ \mathcal{F}^{\geq 1}(\tau), & \text{otherwise.}\end{cases}$$

By combing all of the operations we defined before, the full generalized algorithm can be described as follows.

Algorithm 5 Generalized Detection Algorithm

```
S(0) = ∅;
t = 0;
loop
    t ← t + 1;
    shrink the suspicious set at round t; /*Use alg. 4*/
    S(t) = {F^{≥1}(τ)* | F^{≥1}(τ) ∈ S(t)}; /*Acceleration*/
    for F^{≥1}(τ) ∈ S(t) do
        if |F^{≥1}(τ)| ≤ 1 then
            S(t) = S(t)\{F^{≥1}(τ)};
            remove node in F^{≥1}(τ) from neighbors;
        end if
    end for
end loop
```

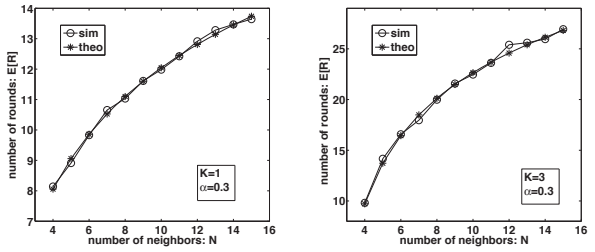
5 Performance Evaluation

In previous sections, we present a set of distributed detection algorithms, as well as the theoretic analysis on their performance. In this section, we validate the analysis and show the effectiveness and efficiency of the algorithms via simulation.

5.1 Performance of Algorithm 1

We first show the performance of Algorithm 1. In this case, malicious nodes only forward polluted packets, i.e., the imitation probability $\delta=0$. Observe that, since no malicious node can evade the detection, i.e., $\mathcal{P}_{fn}(t)=0$, we only show the performance measures of $\mathcal{P}_{fp}(t)$ and $E[\mathcal{R}]$.

We first focus on the performance measure of $E[\mathcal{R}]$, Figure 3 shows the theoretical results and the simulation results. In this simulation, we set the forwarding probability α as 0.3. Figure 3a corresponds to the case of single malicious node and Figure 3b shows the results of the case where three of the N neighbors are malicious. In both figures, the horizontal axis is the number of neighbors N, and the vertical axis is the average number of detection rounds $E[\mathcal{R}]$. From both figures, we can see that the theoretic results fit well with the simulation results. When number of neighbors gets larger, the average number of rounds needed to detect the malicious nodes increases accordingly. Moreover, if multiple malicious nodes exist, then it takes longer time to detect them. However, even in the multiple attackers case, the number of rounds needed for detection is still very small, e.g., it only takes around twenty rounds if the detector has eight neighbors and three of them are attackers. Therefore, Algorithm 1 is very efficient to detect malicious attackers when $\delta = 0$.



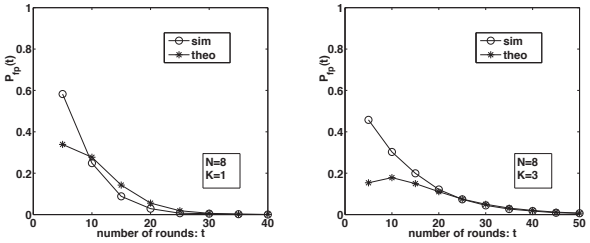
(a) single attacker case (b) multiple attackers case

Fig. 3: Average number of rounds for the case when $\delta = 0$.

Now, let us look at the performance measure of $\mathcal{P}_{fp}(t)$. In this simulation, we set N as eight and the forwarding probability α as 0.3. We also consider the case of single malicious node which is shown in Figure 4a and the case of multiple malicious nodes which is shown in Figure 4b. In both figures, the horizontal axis represents the number of rounds that Algorithm 1 has been run, and the vertical axis shows probability of false positive. The curves with stars correspond to the theoretic results which are derived by Equation (5). From both figures, we can see that the approximated value converges to the accurate value, and when the accurate value is small enough, see < 0.2, the theoretic value is a really good approximation. This convergence shows the effectiveness of the stopping criteria which is designed by using the approximated value of $\mathcal{P}_{fp}(t)$.

5.2 Performance of Algorithm 2

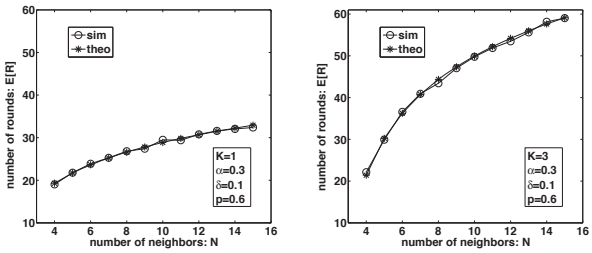
Now, we show the performance of Algorithm 2 which is used to defend against the attack where attackers may imitate legitimate nodes. In this simulation, we still set the forwarding probability α as 0.3. We set δ, the imitation probability, as 0.1, and p, the probability of ignoring a round, as 0.6. The results of performance measure $E[\mathcal{R}]$ are shown in Figure 5. Again, we consider both cases of single attacker and multiple attackers. Figure 5a corresponds to the single attacker case and Figure 5b corresponds to the three attackers case. Firstly, we



(a) single attacker case (b) multiple attackers case

Fig. 4: Probability of false positive $\mathcal{P}_{fp}(t)$.

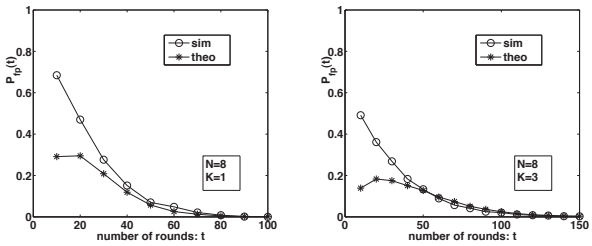
can see that the theoretic results fit well with the simulation results. Secondly, when number of neighbors increases, it takes more rounds to detect the malicious nodes. Moreover, it takes a longer time to detect the attackers when multiple attackers exist. Lastly, by comparing with Figure 3, we can see that it is much more difficult to detect attackers when they imitate the legitimate nodes in some rounds.



(a) single attacker case (b) multiple attackers case

Fig. 5: Average number of rounds for the case when $\delta > 0$.

Figure 6 shows the results of probability of false positive $\mathcal{P}_{fp}(t)$. Figure 6a corresponds to the single attacker case and Figure 6b corresponds to the three attackers case. In both cases, the number of neighbors is set as eight, i.e., $N = 8$. We compare the simulation results with the theoretic results which are computed based on Equation (9). Again, both figures show the convergence of the approximated value, which further show the effectiveness of the stopping criteria.

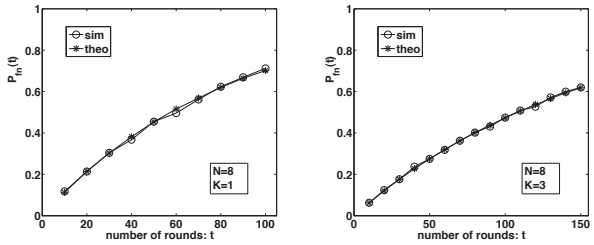


(a) single attacker case (b) multiple attackers case

Fig. 6: Probability of false positive $\mathcal{P}_{fp}(t)$.

Now, let us consider the performance measure of probability of false negative $\mathcal{P}_{fn}(t)$ which is shown in Figure 7. In this simulation, we choose the same parameters as before. Recall

that, since δ is not zero, a malicious node may pretend to be a legitimate node so as to evade the detection. When the number of detection round increases, the chance of evading the detection gets larger, which is shown in both figures. Fortunately, by comparing with Figure 6, even when the number of rounds is large enough such that $\mathcal{P}_{fp}(t)$ is really small, we still have some probability to detect the malicious nodes, i.e., $\mathcal{P}_{fn}(t) \neq 1$. This shows that in every execution of Algorithm 2, we indeed detect some attackers. In other words, it shows the rationality of the enhanced detection algorithm.



(a) single attacker case (b) multiple attackers case

Fig. 7: Probability of false negative $\mathcal{P}_{fn}(t)$.

5.3 Improvement on Probability of False Negative

Using the enhanced detection Algorithm 3 in Section 3.4, we can detect all malicious nodes by repeatedly running the detection algorithm. Table 1 shows the results. In this simulation, we set N as ten and four of them are malicious. Since each time the malicious node can only be detected with probability $1 - \mathcal{P}_{fn}(\bar{t})$, where $\bar{t}$ satisfies $\mathcal{P}_{fp}(\bar{t}) = 0$, i.e., the algorithm runs until all nodes in the suspicious set are malicious. To show the performance, we run the experiments three times (e.g., Exp. A, B and C). Each row in Table 1 corresponds to one experiment outcome. From the table, we can observe that in all experiments, we only need to repeat the detection algorithm a few times to detect all malicious nodes. For example, in the first experiment, two of the four malicious nodes are detected in the first execution. In the second execution, both the remaining two malicious nodes evade the detection. But in the final execution, both malicious nodes are detected. In summary, the enhanced detection algorithm is indeed efficient to detect all malicious nodes.

Experiment	# detected	# detected	# detected
Exp. A	2	0	2
Exp. B	3	1	-
Exp. C	2	2	-

TABLE 1: Performance of the enhanced detection alg.

5.4 Results from System Prototype

To show the effectiveness of the identification and detection of our algorithm, we build a prototype of WMN, which consists of 20 nodes. Each node is equipped with 802.11n transceiver and this WMN is deployed using the MORE protocol and it is network-coding enabled. We consider a particular node 10, which has nine neighbors and they are node 1 to node 9. Node i needs to send packets to node $i+3$ (for $1 \leq i \leq 3$) but since the destination node is not within the transmission range of the sender, all transmissions have to go through node 10. Node 7, 8 and 9 are potential *malicious nodes*, and they probabilistically transmit bogus packets to damage the legitimate transmissions.

We carry out a series of experiments to get the time needed for node 10 to detect all these malicious nodes. In particular, we perform the experiment 200 times. Each time we record the time it takes to detect all three malicious nodes, and then we average all these 200 values. Results are presented in Table 2. As we can see for the table, it takes a short time to detect malicious nodes. For Experiment C, we use the enhanced detection algorithm. We see that it takes around 10 seconds on average to detect node 7, 8 and 9, which send bogus packets so as to create an epidemic spreading.

Experiment	Malicious Nodes	Average Detection Time
Exp. A	node 7	3.60 sec.
Exp. B	node 7 and node 8	6.21 sec.
Exp. C	node 7, node 8 and node 9	10.30 sec.

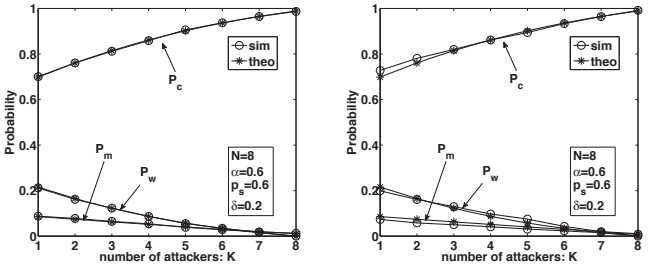
TABLE 2: Average time needed to detect all malicious nodes.

5.5 Performance of the Generalized Algorithm

In this subsection, we validate our theoretic analysis and show the effectiveness and efficiency of the generalized detection algorithm via simulation. We assume that the detector has eight neighbors, i.e., $N = 8$, and at each round, each neighbor forwards with probability 0.6, i.e., $\alpha = 0.6$. At each round, the detector only selects each forwarder for batch verification with probability 0.6, i.e., $p_s = 0.6$. In the simulation, we vary the number of attackers from 1 to N and consider two cases. One is called low imitation case where attackers pretend to be good nodes to forward correct packets with small probability, we choose δ as 0.2 in this case. The other case is called high imitation case where δ is set as 0.6.

We first focus on the low imitation case. To show the effectiveness of the generalized algorithm, we derive the performance measure of probability of correct detection $\mathcal{P}_c$, probability of wrong detection $\mathcal{P}_w$ and probability of miss detection $\mathcal{P}_m$. The simulation and theoretic results are shown in Figure 8. Figure 8a corresponds to the case where the accelerating operation is not performed to speed up the detection and Figure 8b shows the results when the accelerating operation is performed in every round. In both figures, the horizontal axis is the number of attackers in the neighborhood, i.e., K, and we vary it from 1 to N. The vertical axis shows the probabilities. In each figure, we have three groups of curves which show the results of $\mathcal{P}_c$, $\mathcal{P}_w$ and $\mathcal{P}_m$ respectively. For each group of curves, the curve with circles shows the simulation results and the curve with stars shows the theoretic results which are derived by Equation (27). From the figures, we can see that even when the accelerating operation is used, our theoretic results are still very accurate to quantify the

effectiveness of the generalized algorithm. Another conclusion is that the generalized algorithm is very effective to identify the malicious attackers for all cases as the probability of correct detection is always much higher than the probability of wrong detection. Moreover, as the number of attackers increases, the probability of correct detection also increases. The physical meaning is that if most of the neighbors are attackers, then when a forwarder set shrinks to a singleton set, the node in the singleton set is very likely to be a real attacker.

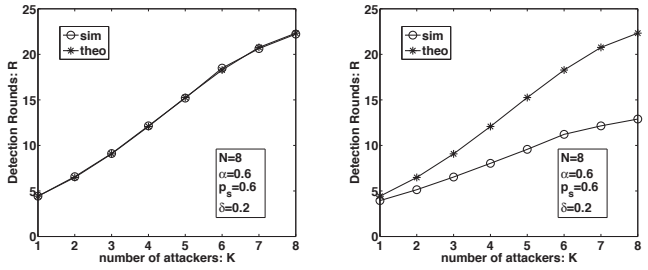


(a) Without acceleration. (b) With acceleration.

Fig. 8: Probability of correct/wrong/miss detection.

Now, let us look at the efficiency of the generalized algorithm. The results are shown in Figure 9. Again, Figure 9a corresponds to the case when accelerating operation is not performed and Figure 9b corresponds to the case when accelerating operation is used to speed up the detection. The horizontal axis is the number of attackers and the vertical axis shows the average number of rounds needed to shrink a forwarder set to a singleton set or an empty set. We also compare the theoretic results which are derived by Equation (26) with the simulation results. Firstly, we can see that the number of rounds needed to shrink the forwarder set is very small, i.e., the generalized algorithm is very efficient to find attackers. Moreover, the smaller number of attackers we have, it is more easier for us to detect. Secondly, although our theoretic analysis on the performance measure of $\mathcal{R}$ does not hold for the case when the accelerating operation is used, it still gives us a good upper bound. Last but not least, we can see that the accelerating operation is very efficient to speed up the detection, it can decrease the number of rounds needed for shrinking the forwarder set to nearly one half. In a word, the generalized algorithm is really efficient in detecting the malicious attackers even if they dominate the neighborhoods.

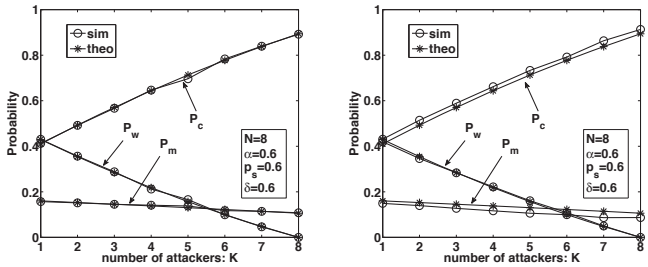
Now, let us show the results when the imitation probability is high, and we set it as 0.6 in the simulation. The results of the performance measures of $\mathcal{P}_c$, $\mathcal{P}_w$ and $\mathcal{P}_m$ are shown in Figure 10, and the results of the performance measure of average number of detection rounds $\mathcal{R}$ are shown in Figure 11. From these figures, we have similar conclusions as the case of low imitation. One thing we have to mention is that when imitation probability gets large, the probability of wrong detection is comparable to the probability of correct detection when the number of attackers is very small, see only one attacker exists. In fact, the physical meaning of this case is reasonable. Intuitively, if only a very small number of attackers exist among a large number of neighbors, and



(a) Without acceleration. (b) With acceleration.

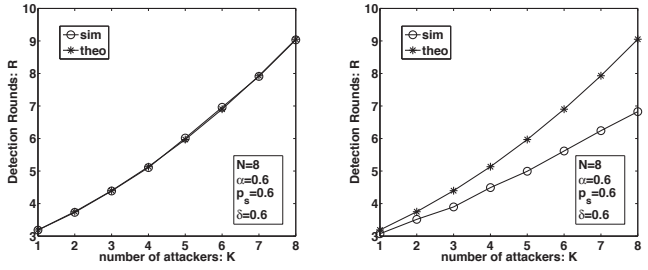
Fig. 9: Average number of detection rounds.

the attackers only occasionally do bad things, then it will be easy for them to entrap other good nodes. In other words, it is hard to distinguish good nodes and the bad nodes which only occasionally behave maliciously. However, by comparing Figure 9 with Figure 11, we can see that choosing higher imitation probability will make the attackers be detected much faster. Moreover, the damage that the attackers can cause is also much smaller if the imitation probability is higher. On the other hand, one can also decrease the selection probability p_s so as to increase the probability of correct detection.



(a) Without acceleration. (b) With acceleration.

Fig. 10: Probability of correct/wrong/miss detection.



(a) Without acceleration. (b) With acceleration.

Fig. 11: Average number of detection rounds.

6 CONCLUSION

In this paper, we present a set of fully distributed defense and detection algorithms to address the pollution attack problem in wireless mesh networks which are configured with

network coding enabled opportunistic routing protocol. The contribution of this paper is on how to effectively discover the malicious nodes *without modifying existing routing protocol and packets verification scheme*, then isolate them from the network so as to defend against the pollution attack. We consider both cases where (1) the malicious nodes always forward polluted packets, and (2) the malicious nodes may pretend to be legitimate nodes and forward valid packets so as to evade the detection. We also propose a general detection algorithm which cannot only detect attackers even if cooperative pollution attack is launched, but also speed up the detection. We provide formal analysis to quantify the performance of our detection algorithms, and extensive simulations are provided to validate the theoretic analysis and show the effectiveness and efficiency of the detection algorithms.

REFERENCES

[1] D. Aguayo, J. C. Bicket, S. Biswas, G. Judd, and R. Morris. Link-level Measurements from an 802.11b Mesh Network. In *SIGCOMM*, pages 121–132, 2004.

[2] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung. Network Information Flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, July 2000.

[3] I. Akyildiz and X. Wang. A Survey on Wireless Mesh Networks. *IEEE Radio communication*, 43(9):S23–S30, September 2005.

[4] P. Bahl, R. Chandra, P. P. C. Lee, V. Misra, J. Padhye, D. Rubenstein, and Y. Yu. Opportunistic Use of Client Repeaters to Improve Performance of WLANs. In *CoNEXT '08: Proceedings of the 2008 ACM CoNEXT Conference*, pages 1–12, New York, NY, USA, 2008.

[5] S. Biswas and R. Morris. Opportunistic Routing in Multi-hop Wireless Networks. *SIGCOMM Comput. Commun. Rev.*, 34(1):69–74, 2004.

[6] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading Structure for Randomness in Wireless Opportunistic Routing. In *SIGCOMM '07: Proceedings of the 2007 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 169–180, New York, NY, USA, 2007. ACM.

[7] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris. A High-throughput Path Metric for Multi-hop Wireless Routing. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 134–146, New York, NY, USA, 2003. ACM.

[8] J. Dong, R. Curtmola, and C. Nita-Rotaru. Practical Defenses Against Pollution Attacks in Intra-flow Network Coding for Wireless Mesh Networks. In *WiSec '09: Proceedings of the second ACM conference on Wireless network security*, pages 111–122, 2009.

[9] J. Dong, R. Curtmola, R. Sethi, and C. Nita-Rotaru. Toward Secure Network Coding in Wireless Networks: Threats and Challenges. *Secure Network Protocols, 2008.*

[10] C. Gkantsidis, W. Hu, P. Key, B. Radunovic, P. Rodriguez, and S. Gheorghiu. Multipath Code Casting for Wireless Mesh Networks. In *CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference*, pages 1–12, New York, NY, USA, 2007. ACM.

[11] T. Ho, B. Leong, R. Koetter, M. Medard, M. Effros, and D. Karger. Byzantine Modification Detection in Multicast Networks with Random Network Coding. *Information Theory, IEEE Transactions on*, 2008.

[12] T. Ho, M. Médard, J. Shi, M. Effros, and D. Karger. On Randomized Network Coding. In *41st Annual Allerton Conference on Communication, Control and Computing, Monticello, IL, USA*, 2003.

[13] S. Katti, H. R. D. Katabi, W. Hu, and M. Medard. The Importance of Being Opportunistic: Practical Netowk Coding for Wireless Environments. In *In Proceedings of 43rd International Conference on Communication, Control and Computing*, 2005.

[14] E. Kehdi and B. Li. Null Keys: Limiting Malicious Attacks Via Null Space Properties of Network Coding. In *INFOCOM 2009, IEEE*, 2009.

[15] M. N. Krohn, M. J. Freedman, and D. Mazières. On-the-Fly Verification of Rateless Erasure Codes for Efficient Content Distribution. In *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, CA, May 2004.

[16] R. K. Lam, D.-M. Chiu, and J. C. S. Lui. On the Access Pricing and Network Scaling Issues of Wireless Mesh Networks. *IEEE Trans. Comput.*, 56(11):1456–1469, 2007.

[17] J. N. Laneman, D. N. C. Tse, and G. W. Wornell. Cooperative Diversity in Wireless Networks: Efficient Protocols and Outage Behavior. *IEEE Transactions on Information Theory*, 50(12):3062–3080, December 2004.

[18] A. Le and A. Markopoulou. Locating Byzantine Attackers in Intra-Session Network Coding Using Spacemac. In *Network Coding (NetCod), 2010 IEEE International Symposium on*, 2010.

[19] J. Le, J. C. S. Lui, and D.-M. Chiu. Dcar: Distributed Coding-Aware Routing in Wireless Networks. *IEEE Transactions on Mobile Computing*, 9:596–608, 2010.

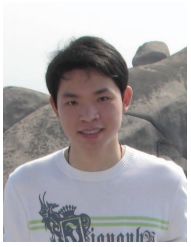
[20] J. Le, J. C. S. Lui, and D.-M. Chiu. On the Performance Bounds of Practical Wireless Network Coding. *IEEE Transactions on Mobile Computing*, 9:1134–1146, 2010.

[21] S.-Y. R. Li, R. W. Yeung, and N. Cai. Linear Network Coding. *IEEE Transaction on Information Theory*, 49(2):371–381, Feb. 2003.

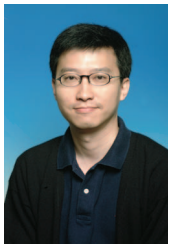
[22] A. Miu, H. Balakrishnan, and C. E. Koksal. Improving Loss Resilience with Multi-radio Diversity in Wireless Networks. In *MOBICOMM*, 2005.

[23] M. Siavoshani, C. Fragouli, and S. Diggavi. On Locating Byzantine Attackers. In *Network Coding, Theory and Applications, 2008.*

[24] S. Vyetrenko, A. Khosla, and T. Ho. On Combining Information-theoretic and Cryptographic Approaches to Network Coding Security Against the Pollution Attack. In *Asilomar'09: Proceedings of the 43rd Asilomar conference on Signals, systems and computers*, pages 788–792, Piscataway, NJ, USA, 2009. IEEE Press.



Yongkun Li was born in China. He received his Bachelor degree from University of Science and Technology of China in 2008. Currently, he is a Ph.D. candidate in the Department of Computer Science and Engineering at The Chinese University of Hong Kong. His research interests lie in the theoretic topics of interactive networks such as P2P networks and online social networks.



John C.S. Lui was born in Hong Kong and is currently a professor in the Department of Computer Science & Engineering at The Chinese University of Hong Kong. He received his Ph.D. in Computer Science from UCLA. When he was a Ph.D student at UCLA, he worked as a research intern in the IBM T. J. Watson Research Laboratory. After his graduation, he joined the IBM Almaden Research Laboratory/San Jose Laboratory and participated in various research and development projects on file systems and parallel I/O architectures. He later joined the Department of Computer Science and Engineering at The Chinese University of Hong Kong. His current research interests are in communication networks, network/ system security (e.g., cloud security, mobile security, etc), network economics, network sciences (e.g., online social networks, information spreading, etc), cloud computing, large scale distributed systems and performance evaluation theory. John serves in the editorial board of IEEE/ACM Transactions on Networking, IEEE Transactions on Computers, IEEE Transactions on Parallel and Distributed Systems, Journal of Performance Evaluation and International Journal of Network Security . John serves as reviewer and panel member for NSF, Canadian Research Council and the National Natural Science Foundation of China (NSFC). John served as the chairman of the CSE Department from 2005-2011. He received various departmental teaching awards and the CUHK Vice-Chancellor's Exemplary Teaching Award. He is also a corecipient of the IFIP WG 7.3 Performance 2005 and IEEE/IFIP NOMS 2006 Best Student Paper Awards. He is an elected member of the IFIP WG 7.3, Fellow of ACM, Fellow of IEEE and Croucher Senior Research Fellow. His personal interests include films and general reading.