Identifying Pollution Attackers in Network-Coding Enabled Wireless Mesh Networks

Yongkun Li
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Email: ykli@cse.cuhk.edu.hk

John C.S. Lui
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Email: cslui@cse.cuhk.edu.hk

***Abstract*—Pollution attack is a severe security problem in network-coding enabled wireless mesh networks (WMNs). Under such form of attack, malicious nodes can easily create an epidemic spreading of polluted packets to deplete network resources. We address this security problem even when the attackers are "*intelligent*" in the sense that they may pretend to be legitimate nodes to *probabilistically* transmit valid packets so as to reduce the chance of being detected. We use the batch verification technique to determine the existence of polluted packets, and propose fully "distributed" and "randomized" detection algorithms to identify the attackers who inject polluted packets, and purge them for future communication. Formal analysis is provided to quantify performance measures of the algorithms, e.g., probability of false positive and probability of false negative, as well as the probability distribution of time needed to identify all malicious nodes. Simulation and system prototype are carried out to show the effectiveness and efficiency of the detection algorithms.**

***Index Terms*—Wireless Mesh Networks; Network Coding; Pollution Attack; Security; Performance Evaluation**

I. Introduction

In recent years, wireless mesh networks (WMNs) have emerged as a promising platform to provide easy Internet access [1], [3], [14]. 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. To improve the performance of WMNs, *opportunistic routing protocol* [4], [8] is used instead. To further improve the spatial reuse, a transmission scheduler based on network coding [16], [17] was proposed. This promising approach can not only improve the end-to-end throughput, but can also reduce packet collision and improve the network capacity. As demonstrated in [16] and systems like COPE [11] and MORE [5], one can achieve the above claims 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. As long as the destination receives enough innovative (or linearly independent) packets, the receiver can decode the received packets and obtain the original data.

However, allowing nodes in a WMN to perform network coding opens the door for *pollution attack* as malicious nodes can inject polluted packets into the network. If an intermediate node is unaware of receiving a polluted packet, it continues to perform the packet encoding and forwards the encoded but polluted packet to its neighbors. Since all nodes participate in coding and packet forwarding, polluted packets will behave like an epidemic and can be easily propagated across the entire network, thereby significantly consume network resource and degrade the performance of legitimate flows. As indicated in [6], pollution attack can be easily launched, and some related work, e.g., [9], [12], [15], [20], [21] address this problem, in particular, on detecting the existence of pollution attack and how to discard polluted packets.

In this paper, we focus on "*identifying*" pollution attackers in WMNs by implementing the idea of shrinking suspicious set [19], then isolate them from the network so as to defend against the pollution attack. Moreover, attackers can be *intelligent* in the sense that they can choose to forward polluted packets, or they can choose to forward valid packets. The rationale that attackers *pretend* to be legitimate nodes is to thwart the detection process so as to reduce the chance of being detected. Contributions of our work are:

- We propose a *randomized* and *fully distributed* identification mechanism: any legitimate node in a WMN can execute our algorithms to identify its malicious neighbors.
- We present a general analytical framework to quantify performance measures of our detection algorithms.
- We validate our analytical model via extensive simulations as well as system prototype, and show the effectiveness and efficiency of the detection algorithms.

The outline of the paper is as follows. In Section II, we briefly provide the necessary background on network coding and batch verification. In Section III, we present the detection methodology in detail and also provide the analysis on its performance measures. We validate our analytical model and present experiment results in Section IV. Section V concludes.

II. Background on Network Coding and Batch Verification

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 most commonly used architecture of WMNs, there is a backbone network which consists of mesh routers. In this paper, we focus on the backbones of WMNs and use network-coding enabled opportunistic routing protocol, which is commonly configured in realistic WMNs to improve network throughput and spatial reuse.

Now, let us provide a brief background on network coding [2], [10], [18]. When a source disseminates a file to destinations, it first breaks up the file into multiple generations. Each generation is further divided into n packets, which are usually referred to as *native packets*. Each packet is composed of m codewords, each of which is regarded as an element in a finite field 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 F_q, i.e., as a column vector with m symbols: $\overrightarrow{p_i} = (p_{1i}, p_{2i}, ..., p_{mi})^\top, p_{ji} \in F_q$. When the 802.11 MAC is ready to send a packet, the source creates a random linear combination of the n native packets, then transmits the coded packet. Formally, a *coded packet* is $(\overrightarrow{e_j}, \overrightarrow{c_j})$ where $\overrightarrow{e_j} = \sum_i c_{ji}\overrightarrow{p_i}$. Each c_{ji} is a random coefficient and we call $\overrightarrow{c_j} = (c_{j1}, c_{j2}, ..., c_{jn})$ a code vector.

For an intermediate node, when it is ready to transmit and has received multiple packets, it first encodes the received packets, then forwards the result packet after encoding. 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. We say a coded packet $(\overrightarrow{e_j}, \overrightarrow{c_j})$ is valid/correct if and only if $\overrightarrow{e_j}$ can be represented as a linear combination of the native packets by using the code vector $\overrightarrow{c_j}$, i.e., $\overrightarrow{e_j} = \sum_i c_{ji}\overrightarrow{p_i}$ holds. Otherwise, we call it a polluted/bogus packet. As shown in [7], the threat of pollution attack is very severe.

One common approach to filter out polluted packets is to perform hash verification by using homomorphic hash functions [13]. However, due to the high computational cost of modular exponentiation, verifying every received packet is impractical for wireless systems [6]. Therefore, batch verification must be considered when design practical verification schemes. Specifically, to verify a set of l coded packets $(\overrightarrow{e_j}, \overrightarrow{c_j})$ $(j = 1, 2, ..., l)$, we only check whether the random linear combination of the l packets, $(\overrightarrow{e}, \overrightarrow{c})$, is correct or not. If it is correct, then all of the l packets are correct. Otherwise, at least one of the l packets is polluted and we have *no knowledge* of which are the polluted packets. In this paper, if the result of the batch verification shows that the coded packet is correct, we call it the batch verification matches, otherwise, we call it the batch verification does not match.

III. **Detection Methodology**

In this section, we present the detection algorithms to identify pollution attackers in a network-coding enabled WMN, as well as the performance measures of the detection algorithms.

A. 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 them in an asynchronous fashion to identify attackers among its neighbors. In this paper, we only focus on one legitimate node, say node i, and describe its operations to identify malicious neighbors. Let $\mathcal{N}^i$ be the set representing the neighbors of node i, and we assume that $|\mathcal{N}^i| = N$. Among these N neighbors, there can be *multiple malicious* nodes.

For a malicious node, when it is ready to broadcast a packet to other nodes, it may choose one of the following actions:

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

The reason why a malicious node may choose to imitate as a legitimate node is to thwart the detection so as to reduce the chance of being detected. On the other hand, for any legitimate node, to guarantee the correctness of received packets and reduce the end-to-end delay, *batch verification* is used, and we assume that legitimate nodes only forward valid packets, i.e., they discard packets which do not pass the batch verification, so as to prevent the epidemic spreading of polluted packets.

We define the duration of node i receiving packets 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 i. At round t, some neighbors of node i may forward packets to it and others may not. We define $\mathcal{F}(t)$ as the set of neighbors of node i which forward innovative packets to it at round t, and $\bar{\mathcal{F}}(t)$ as the set of neighbors which do not forward innovative packets to it at round t. Obviously, we have $\mathcal{N}^i = \mathcal{F}(t) \cup \bar{\mathcal{F}}(t)$.

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

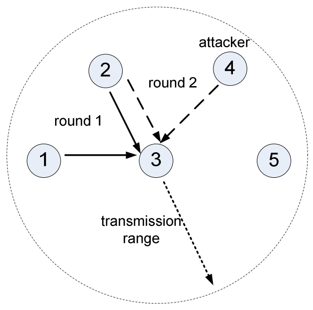


Fig. 1: Illustration of the detection mechanism.

Consider the example in Figure 1 in which node 4 is an attacker, we focus on node 3 and take it as a detector. Node 1, node 2, node 4 and node 5 are all within the transmission range of node 3, i.e., $\mathcal{N}^3 = \{1, 2, 4, 5\}$. We initialize the suspicious set at round 0 as $\mathcal{S}(0) = \mathcal{N}^3$. In the first round, if node 1 and node 2 forward valid packets to node 3, then node 3 knows that node 1 and node 2 must be legitimate nodes after performing batch verification, or the attacker exists in the suspicious set $\mathcal{S}(1) = \{4, 5\}$. 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 will not match. Therefore, node 3 knows that node 2 and node 4 are suspicious nodes and it can shrink the suspicious set as $\mathcal{S}(2) = \mathcal{S}(1) \cap \{2, 4\} = \{4\}$. In other words, node 3 identifies the attacker by shrinking the suspicious set.

Besides developing algorithms to implement the above idea, we also provide theoretic analysis on their performance. In particular, we quantify the performance measures:

- $\mathcal{P}_{fn}(t)$, probability of false negative until round t,
- $\mathcal{P}_{fp}(t)$, probability of false positive until round t, and
- $E[\mathcal{R}]$, expected number of rounds needed to detect the attackers with high accuracy.

The first two performance measures quantify the *accuracy* of the detection algorithms, while the third one quantifies the *efficiency*. Precisely, $\mathcal{P}_{fn}(t)$ is defined as the probability of a malicious node being wrongly removed from the suspicious set 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. $\mathcal{R}$ denotes the number of detection rounds until all nodes in $\mathcal{S}(t)$ are malicious nodes. In the following, we separate the analysis into two cases to illustrate our detection algorithms:

Case 1: *malicious nodes will not imitate the action of a legitimate node, or the imitation probability* $\delta = 0$;

Case 2: *malicious nodes will imitate the action of a legitimate node to reduce the chance of being detected, or* $\delta > 0$.

Note that detection in the second case is more challenging since attackers are trying to thwart the detection process. However, our detection algorithms can still effectively identify them even if they may disguise as legitimate nodes.

B. Case 1: Attackers with Imitation Probability $\delta = 0$

In this subsection, we consider the case when $\delta = 0$, i.e., when a malicious node attempts to transmit, it always broadcasts polluted packets. We only focus on a particular node i and take it as a detector. We initialize the suspicious set $\mathcal{S}(0)$ as $\mathcal{N}^i$. At each round, nodes in $\mathcal{N}^i$ are classified into different types according to their behaviors. Based on this classification, the suspicious set $\mathcal{S}(t)$ shrinks and eventually, it only contains malicious nodes, then node i can claim that it has identified its malicious neighbors. Formally, our detection algorithm at round t can be described as follows.

Algorithm A: Detection Algorithm for $\delta = 0$

```
if (the batch verification matches): $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1) \cap \bar{\mathcal{F}}(t)$;
else:                                $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1)$;
```

The rationale of Algorithm A is as follows. At round t, node i performs batch verification, if it matches, i.e., the coded packet is valid, then malicious nodes cannot be in the forwarding set $\mathcal{F}(t)$ because they only forward polluted packets. On the other hand, if there is a mismatch of verification, it may be caused by two cases: (1) all malicious nodes forward polluted packets, or (2) only some of the malicious nodes forward polluted packets but others do not perform packet forwarding. Therefore, node i cannot classify its neighboring nodes and the suspicious set $\mathcal{S}(t)$ remains unchanged.

Since fairness is a built-in feature in the medium access control (MAC) protocol in wireless networks, a node cannot monopolize the wireless resource by repeatedly sending packets, i.e., when the communication channel is free, all nodes which have backlog packets will compete for the channel. Therefore, a successful forwarding does not depend on whether a node is malicious or not, but rather depends on whether it has packets to be forwarded or not. We define variable α to represent the probability that a neighbor of node i performs forwarding at each round, which is called *forwarding probability*. Note that, a generation only contains 32 independent packets in the common setting under MORE, and a node may perform multiple verifications during the period of transmitting one generation. Furthermore, when a node is ready to transmit, it not only performs one transmission, but performs multiple transmissions. Therefore, α is less than one for most cases, i.e., the neighbors of node i can be distinguished such that the suspicious set can shrink.

To quantify the performance measures of Algorithm A, we assume that the number of malicious neighbors of node i is k ($k \geq 1$). Moreover, since the suspicious set cannot be shrunk in every round, we define those rounds in which the suspicious set shrinks as *detectable rounds* and use random variable $\mathcal{D}$ to indicate the number of detectable rounds. The performance measures of Algorithm A are stated in Lemma 1.

Lemma *1:* In the case when $\delta = 0$, after Algorithm A runs for t rounds, $\mathcal{P}_{fn}(t) = 0$, $\mathcal{P}_{fp}(t) = \frac{|\mathcal{S}(t)|-k}{|\mathcal{S}(t)|}$ and $\mathcal{R}$ follows the distribution of $P(\mathcal{R} = r) = \sum_{d=1}^{r} \binom{r-1}{d-1} (1-\alpha)^{kd} [1-(1-\alpha)^k]^{r-d} P(\mathcal{D} = d)$.

Proof: According to Algorithm A, since no malicious node is wrongly removed from $\mathcal{S}(t)$, the probability of false negative is simply zero, i.e., $\mathcal{P}_{fn}(t) = 0$.

Note that, all of the k malicious nodes stay in the suspicious set $\mathcal{S}(t)$. Furthermore, all of the nodes in the suspicious set are taken as malicious nodes at the end of the detection process. Therefore, $\mathcal{P}_{fp}(t) = \frac{|\mathcal{S}(t)|-k}{|\mathcal{S}(t)|}$. Since $\lim_{t\to\infty} |\mathcal{S}(t)| = k$ as $\mathcal{P}_{fn}(t) = 0$, we have $\lim_{t\to\infty} \mathcal{P}_{fp}(t) = 0$.

To derive the distribution of $\mathcal{R}$, we first derive the distribution of $\mathcal{D}$. Note that, in a detectable round, a legitimate node is removed from the suspicious set only when it forwards at that round, which happens with probability α. Therefore,

$$\begin{aligned} P(\mathcal{D} \leq d) &= P(\text{after } d \text{ detectable rounds, no legitimate} \\ &\qquad \text{node remains in the suspicious set } \mathcal{S}) \\ &= \left(1 - (1-\alpha)^d\right)^{N-k}. \end{aligned} \tag{1}$$

Using detection Algorithm A, a detectable round only happens when no attacker forwards packets to node i, so $p_d = (1-\alpha)^k$. On the other hand, given the number of detectable rounds $\mathcal{D}$, the conditional distribution $P(\mathcal{R} = r|\mathcal{D} = d)$ is a negative binomial distribution, so we have:

$$\begin{aligned} P(\mathcal{R} = r) &= \sum_{d=1}^{r} P(\mathcal{D} = d)P(\mathcal{R} = r|\mathcal{D} = d) \\ &= \sum_{d=1}^{r} \binom{r-1}{d-1} (p_d)^d (1-p_d)^{r-d} P(\mathcal{D}=d). \quad (2) \end{aligned}$$

By replacing $P(\mathcal{D} = d)$ based on Equation (1), we have the distribution of $\mathcal{R}$, and $E[\mathcal{R}]$ can be easily computed via $E[\mathcal{R}] = \sum_{r=1}^{\infty} rP(\mathcal{R} = r)$. ■

C. Case 2: Attackers with Imitation Probability $\delta > 0$

Let us consider a more interesting case where a malicious node may imitate as a legitimate node by forwarding valid packets with probability δ ($\delta > 0$). Under this situation, if the verification does not match, node i knows it must have received some polluted packets. However, due to the existence of multiple attackers, node i can not be certain whether all attackers are in the forwarding set or not. On the other hand, if the verification matches, node i still faces with the problem of accurately distinguishing the malicious attackers since they may pretend to be legitimate nodes. Note that, the goal of the malicious nodes is to reduce the system performance or to damage the system by injecting polluted packets. Obviously, the action of imitation violates this objective. Therefore, the imitation probability δ cannot be too large. Based on this fact, we propose the following "*randomized detection algorithm*".

Algorithm B: Randomized Detection Algorithm for $\delta > 0$

```
if (the batch verification matches):
    with probability $p$:      $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1)$;
    with probability $1-p$:    $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1) \cap \bar{\mathcal{F}}(t)$;
else: $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1)$;
```

We use a 0-1 random variable $d(t)$ to indicate whether round t is detectable or not, $d(t)$ equals to one if round t is a detectable round and zero otherwise. In algorithm B, a round is detectable if and only if the verification matches and that round is not ignored. We call $d(t)$ the detectable indicator. At round t, node i knows the forwarding set $\mathcal{F}(t)$ which contains its neighbors that forwarded packets to it. Therefore, node i has two parameters which are the forwarding set $\mathcal{F}(t)$ and the detectable indicator $d(t)$ to record the information it obtains at round t. We define the state of node i at round t as $s(t) = (\mathcal{F}(t), d(t))$. Given the state of node i at round t, the conditional probability of a neighbor (malicious or legitimate) of node i forwarding at that round is simply $|\mathcal{F}(t)|/N$, and we let $\alpha(t) = |\mathcal{F}(t)|/N$. The collection of all states in t rounds compose the detection history of node i until round t, which is denoted as $\mathcal{H}(t)$, i.e., $\mathcal{H}(t) = (\mathcal{F}(1), d(1)), (\mathcal{F}(2), d(2)), ..., (\mathcal{F}(t), d(t))$. Based on the detection history $\mathcal{H}(t)$, the performance measures of Algorithm B can be derived, which are stated in Lemma 2.

Lemma *2:* In the case when $\delta > 0$, after Algorithm B runs for t rounds, $\mathcal{P}_{fn}(t) = 1 - \prod_{\tau=1,d(\tau)=1}^{t} \frac{1-\alpha(\tau)}{1-\alpha(\tau)+\alpha(\tau)\delta}$ and $\mathcal{P}_{fp}(t) = \frac{(N-k)\prod_{\tau=1,d(\tau)=1}^{t}(1-\alpha(\tau))}{(N-k)\prod_{\tau=1,d(\tau)=1}^{t}(1-\alpha(\tau)) + k\prod_{\tau=1,d(\tau)=1}^{t}\left(1-\frac{\alpha(\tau)\delta}{1-\alpha(\tau)+\alpha(\tau)\delta}\right)}$.

Proof: Based on the randomized Algorithm B, when a malicious node pretends to be a legitimate node (forwarding valid packets) in a detectable round, it is removed from the suspicious set forever, which means that it evades the detection. Therefore, the probability of false negative is:

$$\begin{aligned} \mathcal{P}_{fn}(t) &= P(\text{after } t \text{ rounds, a malicious node} \\ &\quad \text{is not in the suspicious set } \mathcal{S}(t) \mid \mathcal{H}(t)) \\ &= 1 - \prod_{\tau=1,d(\tau)=1}^{t} \frac{1-\alpha(\tau)}{1-\alpha(\tau)+\alpha(\tau)\delta}. \quad (3) \end{aligned}$$

On the other hand, if Algorithm B is not executed for sufficient number of rounds, it is possible that some legitimate nodes still remain in the suspicious set $\mathcal{S}(t)$, and they are wrongly claimed as attackers. Observe that, a node is removed from the set $\mathcal{S}(t)$ only when it forwards valid packets in some detectable round τ. For a malicious node, this probability is $\frac{\alpha(\tau)\delta}{1-\alpha(\tau)+\alpha(\tau)\delta}$, and we denote it as $P_M(\tau)$. Similarly, for a legitimate node, the probability is just $\alpha(\tau)$, and we denote it as $P_L(\tau)$. We have:

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

By substituting $P_L(\tau)$ and $P_M(\tau)$, we have Lemma 2. ■

The performance measure of $E[\mathcal{R}]$ of Algorithm B is similar with the results stated in Lemma 1, so we omit it here. The only parameter we need to recalculate is p_d, the probability of a round being detectable. When algorithm B is employed, this probability changes to be $p_d = (1 - \alpha + \alpha\delta)(1 - p)$.

D. Detection Algorithm to Enhance $\mathcal{P}_{fn}(t)$

In the case where the imitation probability $\delta > 0$, when algorithm B runs for sufficient number of rounds such that all nodes in the suspicious set are attackers, the probability of false negative does not converge to zero. In other words, some malicious attackers evade the detection. To improve the detection accuracy, we develop an enhanced algorithm. The idea is as follows. After running the detection Algorithm B for sufficient number of rounds, all nodes in the suspicious set $\mathcal{S}(t)$ are malicious with very high probability. One can first remove them (i.e., blacklist them for further data exchange), then repeat the detection process again. After executing the detection process multiple times, one can be certain in removing all malicious nodes from the neighbor list.

Algorithm C: Enhanced Detection Algorithm

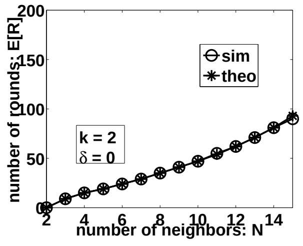
```
repeat{
    do {
        if (the batch verification matches) {
            with probability $p$ : $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1)$;
            with probability $1-p$ : $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1) \cap \bar{\mathcal{F}}(t)$;}
        else: $\mathcal{S}(t) \leftarrow \mathcal{S}(t-1)$;}
    while ($\mathcal{S}(t)$ contains legitimate node)
    remove nodes in $\mathcal{S}(t)$ from the neighbor list $\mathcal{N}^i$; }
until (all malicious nodes in $\mathcal{N}^i$ are detected)
```

IV. Performance Evaluation

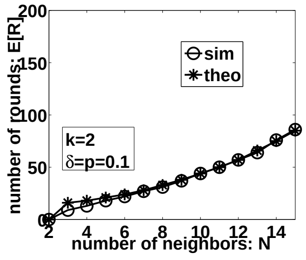
In this section, we validate the analysis by comparing the theoretical results with simulation results. In the simulation, at each round, we randomly choose the forwarding probability α from $[0,1]$ to simulate the forwarding process. We also consider two cases: (1) $\delta = 0$, and (2) $\delta > 0$. We first validate the performance measure of $E[\mathcal{R}]$, then consider probability of false negative and probability of false positive.

A. Performance Measure of $E[\mathcal{R}]$

The performance measure of $E[\mathcal{R}]$ is an average view of the number of rounds needed to detect malicious nodes. Figure 2 shows the theoretical results and the simulation results when $\alpha = 0.5$ in all rounds. In both figures, the horizontal axes are the size of the neighboring set, and we assume that two of the N neighbors are malicious, i.e., $k = 2$. The vertical axes show the average number of rounds, $E[\mathcal{R}]$. Figure 2a shows the results in the case when $\delta = 0$ and Figure 2b corresponds to the case when $\delta > 0$. From both figures, we can see that the theoretic results fit well with the simulation results. Moreover, when the size of the neighboring set gets larger, the average number of rounds needed for detection increases accordingly.



(a) $\delta = 0$



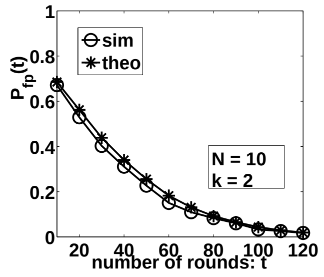
(b) $\delta > 0$

Fig. 2: Average number of rounds, $E[\mathcal{R}]$.

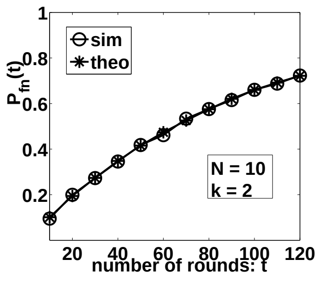
B. Performance Measures of $\mathcal{P}_{fn}(t)$ and $\mathcal{P}_{fp}(t)$

Now, we focus on the performance measures of probability of false negative and probability of false positive. Since the derivation of $\mathcal{P}_{fn}(t)$ and $\mathcal{P}_{fp}(t)$ when $\delta = 0$ is trivial, we only validate the analysis for the case when $\delta > 0$. In the simulation, we set both δ, the imitation probability, and p, the probability of ignoring a round, as 0.1.

Figure 3 shows the simulation results and theoretic results. In both figures, the horizontal axes are the number of rounds t, and the vertical axes represent probability. Figure 3a shows the performance measure of probability of false positive, $\mathcal{P}_{fp}(t)$, and Figure 3b shows probability of false negative, $\mathcal{P}_{fn}(t)$. We assume that node i has ten neighbors, i.e., $N = 10$, and two of them are attackers. Firstly, we can see that the theoretic results fit well with the simulation results. Secondly, when the number of rounds t gets larger, $\mathcal{P}_{fp}(t)$ converges to 0. In other words, if the detection algorithm runs for enough rounds, then all nodes in the suspicious set $\mathcal{S}(t)$ are malicious. However, when the number of detection rounds increases, the probability of attackers evading the detection also gets larger, which is shown in figure 3b. Fortunately, by comparing with Figure 3a, even when the number of rounds is large enough such that $\mathcal{P}_{fp}(t) = 0$, we still have some chances to detect the malicious nodes, i.e., $\mathcal{P}_{fn}(t) < 1$. This shows the rationality of our enhanced detection algorithm C.



(a) probability of false positive



(b) probability of false negative

Fig. 3: Probability of false positive and probability of false negative for the case when $\delta > 0$.

C. Effectiveness of the Enhanced Algorithm

Using the enhanced detection algorithm C in Section III-D, we can identify all malicious nodes by repeatedly running the detection algorithm. Table I shows the results. In this simulation, we have $|\mathcal{N}^i| = 10$ 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 $\mathcal{P}_{fp}(\bar{t}) = 0$, we repeat the detection experiment three times (e.g., Exp. A, B and C) to show the effectiveness of the enhanced algorithm. Each row in Table I corresponds to an experiment outcome. We can observe that in all experiments, we only need to repeat the detection algorithm a few times to detect all malicious nodes, e.g., in the first experiment, two of the four malicious nodes are detected in the first execution, the remaining two malicious nodes evade the detection in the second execution, but in the third execution, the last two malicious nodes are both detected. In summary, the enhanced detection algorithm can effectively identify all malicious nodes.

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

TABLE I: Effectiveness of detecting all attackers.

D. *Results from System Prototype*

To show the effectiveness of identification and detection of our algorithms, 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 so as to damage the legitimate transmissions.

We carry out a series of experiments to get the time needed for node 10 to detect all malicious nodes. In particular, we perform the experiment 200 times. Each time we record the time it takes to detect all malicious nodes, then we average all these 200 values. Results are presented in Table II. As we can see from the table, it takes a very short time to detect malicious nodes. For Experiment C, we use the enhanced detection Algorithm C. 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 II: Experimental results: average time needed to detect all malicious nodes.

V. **Conclusion**

In this paper, we present a set of fully distributed algorithms to address the pollution attack problem in network-coding enabled WMNs. The contribution of this paper is on *how to effectively identify 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 of (1) malicious nodes always forwarding polluted packets, and (2) smart malicious nodes which may pretend to be legitimate nodes and forward valid packets from time to time so as to evade the detection. We also provide formal analysis on quantifying the performance measures of the detection algorithms, and validate them via extensive simulations and system prototype.

Acknowledgement: this research is supported in part by the UGC, Project No. AoE/E-02/08 and RGC Project 415309.

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), 2000.

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

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

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

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

[19] Y. Li and J. C. Lui. Stochastic Analysis of a Randomized Detection Algorithm for Pollution Attack in P2P Live Streaming Systems. *Performance Evaluation*, 67(11):1273 – 1288, 2010. Performance 2010.

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

[21] 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.