On Detecting Malicious Behaviors in Interactive Networks: Algorithms and Analysis

Yongkun Li, John C.S. Lui
Department of Computer Science & Engineering, The Chinese University of Hong Kong
Email: {ykli, cslui}@cse.cuhk.edu.hk

Abstract—**Interactive networks are vulnerable to various attacks due to the existence of malicious nodes which do not comply with the network protocol so as to achieve their own purposes. For example, in a peer-to-peer (P2P) streaming system, since each peer needs to participate in uploading data to other peers, malicious peers may choose to upload bogus data so as to damage the playback and degrade the watching experience of normal peers in the system. This is known as pollution attack in P2P networks, and it can cause severe impact to the performance of P2P streaming systems. Other examples include pollution attack in wireless mesh networks (WMNs) where malicious nodes forward modified and polluted packets to other nodes, and the shill attack in online social networks (OSNs) where malicious users give wrong recommendations to others so as to mislead their purchases. In this paper, we propose a general and fully distributed detection framework which can be executed by each legitimate node in an interactive network to identify its malicious neighbors. To illustrate the effectiveness and the efficiency of our detection framework, we apply it to three realistic applications: P2P streaming networks, WMNs and OSNs, and show how to defend against the attacks launched by malicious nodes. We also quantify the performance of our detection algorithms and validate our analysis via extensive simulations.**

Index Terms—**Interactive Networks; Security; Malicious Behavior Detection**

I. Introduction

Many network services consist of a large set of independent nodes, and these nodes are required to follow some rules (or protocols) to cooperate so as to achieve a given network functionality. To realize such network service, every nodes must communicate or provide local services with a subset of other nodes which are called neighbors, e.g., upload packets to neighbors, download packets from neighbors and forward packets for neighbors and so on. To guarantee the correct functionality of the network service such that every node can get service with desired performance, nodes must follow the predefined protocols when they participate in the communication with their neighbors. However, there are number of scenarios that nodes may not want to follow the protocol. For one, some nodes may violate the rules and behave maliciously so as to obtain some personal gain. For others, it is possible that a third party may inject some mis-behaving nodes so as to disrupt the network service to gain a competitive edge in a commercial market.

Malicious nodes exist in many interactive network services where nodes must communicate with each other to achieve some functionality, for example, in P2P networks, peers must cooperate with others and participate in uploading and downloading. Specifically, when a peer wants to playback a piece of video file, it must first download all required sub-pieces from its neighbors who have these data. On the other hand, if its neighbors request some sub-pieces that it possesses, it also needs to upload the sub-pieces to its neighbors. Therefore, every node follows this rule to participate in uploading and downloading so that it can get the desired data for playback. However, some nodes may not follow these cooperative rules, e.g., they may upload polluted sub-pieces to their neighbors instead of uploading correct sub-pieces, and this is the well-known *pollution attack* in P2P networks [18]. There are many causes for pollution attack. One of them is that a P2P software is mis-configured and it uploads polluted information to other peers. Another possibility is that another P2P service provider hires a pollution agent to carry out such attack to its competitors. As reported in [6], pollution attack can reduce the number of legitimate peers by as much as 85%. Furthermore, the bandwidth used in exchanging the polluted data may deplete the network resource, and lowering the efficiency.

Another example of malicious behavior in interactive networks happens in WMNs which have network coding opportunistic routing enabled. In these networks, every node operates not only as a host but also as a router, and it forwards packets for other nodes that are not in the direct transmission range of their destinations. Since network coding is enabled, intermediate nodes first mix (or encode) received packets and then forward the coded packet to other nodes. However, malicious nodes may inject polluted packets in the system as they are allowed to modify received packets (since this is part of the network coding operation). This is also known as the *network coding pollution attack* in WMNs.

Last but not least, malicious nodes can also be found in virtual networks like OSNs where users frequently interact with their friends and participate in providing and receiving recommendations. Specifically, users often seek or receive recommendations from their friends before they do any purchase. On the other hand, when one buys a product, she may make recommendations to her friends such that they may be influenced to do further purchase. However, malicious users may give *wrong* or *misleading* recommendations to their neighbors. We call such attack the *shill attack* in OSNs. There are many reasons to motivate such kind of attack, e.g., firms may hire some users in an OSN to promote their products, worse yet, they may even consider paying users to provide misleading recommendations on their competitors' products.

In this paper, our goal is to develop a "*general theory*" to detect and identify malicious nodes in interactive networks. Once we identify these malicious nodes, one can block them for further communication so as to reduce their impact to the normal network service. Specifically, we develop a *general and fully distributed detection framework* which can be easily implemented by each node to identify its malicious neighbors. To illustrate the theory and framework, we apply it to the three applications we mentioned above and show how to defend against the pollution attack in P2P networks, WMNs, and OSNs. The main contributions of our work are:

- We develop a general and fully distributed detection framework which can be executed by every legitimate node so as to identify its malicious neighbors.
- We apply our general detection framework to a large class of interactive networks to identify malicious nodes so as to defend against malicious attack.
- We show the effectiveness of our framework by deriving three performance measures: probability of false negative, probability of false positive and the expected detection time to quantify the performance of our detection algorithms under different applications.

The organization of this paper is as follows. In Section II, we present a general detection framework and define three performance measures. We apply the detection framework to P2P streaming systems, WMNs and OSNs to defend against pollution attack and shill attack in Section III, Section IV and Section V respectively. In Section VI, we run simulation to validate our model and analysis, and Section VII concludes.

II. General Detection Framework

We consider a network consisting of a set of independent nodes, and each node interacts with a subset of other nodes which are called its neighbors. Nodes are required to obey some predefined rules for cooperation so as to achieve some network functionality. However, some nodes may not follow the rules to collaborate, and conversely, they carry disruptive operations when interact with their neighbors, and we call such nodes *the malicious nodes*. Correspondingly, the nodes which follow the rules are called *good nodes* or *legitimate nodes*. The purpose of this paper is to identify the malicious nodes and block them for further communication so as to guarantee the functionality of the network. To achieve it, we design a fully distributed detection algorithm which can be executed by every good node to identify its malicious neighbors.

Let us focus on a particular good node, say node i. In the rest of this paper, we refer to node i as the detector. Define $\mathcal{N}^i$ as the neighboring set of node i, i.e., node i only interacts with nodes in $\mathcal{N}^i$. Since only two types of nodes exist in the network, good nodes and malicious nodes, every node j in $\mathcal{N}^i$ must be either good or malicious, we use t_j to denote its type, and $t_j \in \{good, malicious\}$. We model the interaction between node i and its neighbors as a repeated process and time proceeds in rounds. At each round, node i may interact with some of its neighbors in $\mathcal{N}^i$ to achieve some objective. We call the set of such neighbors *the active set* and use $\mathcal{A}(t)$ to denote the active set at round t. Intuitively, every node in $\mathcal{A}(t)$ contributes to node i at round t. However, if a malicious node exists, node i may not achieve its objective due to the disruptive operations carried out by the malicious node. Formally, we define the payoff of node i for each interaction with its neighbor j as follows.

$$U_{ij} = \begin{cases} 1, & \text{if } t_j = good, \\ -\infty, & \text{if } t_j = malicious. \end{cases} \tag{1}$$

In a realistic system, nodes do not have global information. Specifically, node i can only observe the behaviors of its neighbors but not their types, i.e., node i does not know the payoff gained from each individual neighbor. In this work, we assume that node i can *measure* the total payoff gained at each round, which is the sum of the payoff gained from each individual neighbor in the active set. We define $U(t)$ as the total payoff at round t. Formally, we have

$$U(t) = \sum\nolimits_{j \in \mathcal{A}(t)} U_{ij}. \tag{2}$$

Since U_{ij} is either 1 or $-\infty$ based on Equation (1), $U(t)$ can be either $|\mathcal{A}(t)|$ or $-\infty$. If $U(t)$ equals to $|\mathcal{A}(t)|$, then it means that all U_{ij} are equal to one. In other words, all nodes in the active set at time t must be good nodes. On the other hand, if $U(t)$ is equal to $-\infty$, then at least one U_{ij} equals to $-\infty$, i.e., the active set $\mathcal{A}(t)$ contains at least one malicious node. We define *secure set* as the set of nodes which are identified as good nodes, and use $\mathcal{S}(t)$ to denote it at round t.

$$\mathcal{S}(t) = \begin{cases} \mathcal{A}(t), & \text{if } U(t) = |\mathcal{A}(t)|, \\ \emptyset, & \text{if } U(t) = -\infty. \end{cases} \tag{3}$$

Now, we can describe the general framework of our detection algorithms. Observe that, initially, there maybe several malicious nodes existing in the neighboring set $\mathcal{N}^i$. Furthermore, node i does not know which neighbors are malicious nodes and which neighbors are good nodes. In other words, every node in the neighboring set is potentially malicious. Define $\mathcal{M}(t)$ as the set of nodes which are potentially malicious until round t, i.e., all neighboring nodes that are not in $\mathcal{M}(t)$ are considered as good nodes by node i. Initially, we have $\mathcal{M}(0) = \mathcal{N}^i$. We call $\mathcal{M}(t)$ *the malicious set* at time t. To identify the malicious nodes in $\mathcal{N}^i$, observe that at each round, node i is sure that all nodes in the secure set $\mathcal{S}(t)$ are good nodes and they can be removed from the malicious set. If we execute our algorithm for sufficient number of rounds, one can expect that all good nodes are removed from the malicious set and $\mathcal{M}(t)$ only contains malicious nodes. Formally, we have

$$\mathcal{M}(t) = \mathcal{M}(t-1) - \mathcal{S}(t). \tag{4}$$

The implementation of the above idea is stated as follows.

Alg. A: General Detection Framework for Node i

```
$t \leftarrow 0$;
$\mathcal{M}(0) = \mathcal{N}^i$;
**while** ($\mathcal{M}(t)$ contains good nodes){
        /* shrink the malicious set */
        $t \leftarrow t+1$;
        $\mathcal{M}(t) \leftarrow \mathcal{M}(t-1) - \mathcal{S}(t)$;
}
remove all nodes in $\mathcal{M}(t)$;
```

We like to point out that, since the detection algorithm is fully distributed, i.e., every good node can execute it to identify its malicious neighbors. Therefore, all malicious nodes in the whole network can be identified.

One technical issue that we need to answer is how *efficient* is the proposed detection framework? Here, we use the following performance measures to quantify our detection framework:

- $\mathcal{P}_{fn}(t)$, 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 for detection.

The first two performance measures quantify the *accuracy* of the detection framework, while the third one quantifies the *efficiency*. Formally, $\mathcal{P}_{fn}(t)$ is defined as the probability that a malicious node is wrongly regarded as a good node, which is actually the probability that a malicious node is wrongly removed from the malicious set $\mathcal{M}(t)$ since only nodes in the malicious set are regarded as malicious nodes. On the other hand, $\mathcal{P}_{fp}(t)$ is defined as the probability that a good node is wrongly regarded as a malicious node. Since we regard all nodes in the malicious set $\mathcal{M}(t)$ as malicious nodes, $\mathcal{P}_{fp}(t)$ is the probability that a good node remains in the malicious set $\mathcal{M}(t)$. Lastly, the random variable (r.v) $\mathcal{R}$ denotes the number of detection rounds until no good node exists in $\mathcal{M}(t)$.

In the following sections, we illustrate how to use the general detection framework to identify malicious nodes in various interactive networks. Due to page limit, we only present the results for the case that malicious nodes always behave maliciously in P2P streaming systems and OSNs. We also studied the case of intelligent attack where malicious nodes may pretend as good nodes in following the network protocol. The rational is that malicious nodes may want to confuse the detector so as to avoid the detection. Please refer to [14]–[16] for more details.

III. Case Study: Peer-to-Peer Streaming Systems

In this section, we illustrate how to apply the general detection framework to P2P streaming systems so as to defend against pollution attack. First, we briefly introduce the background on P2P streaming systems and the pollution attack problem, then we present a fully distributed detection algorithm which is based on our general detection framework to identify malicious nodes in P2P streaming systems. We also derive the three measures we defined before to quantify the performance and efficiency of our detection algorithm.

A. Background on P2P Streaming Systems

P2P streaming systems consist of a large set of peers where each peer downloads desired data from other peers, and also uploads data it has to other peers. The server of a P2P streaming system which generates video contents first divides a video file into non-overlapping chunks, and a chunk is the unit of P2P advertisement. Specifically, each peer maintains a bitmap to record its chunks availability, and peers periodically exchange their bitmaps to inform other peers which chunks they are holding. To balance between communication/memory overheads and playback requirement, a chunk is further divided into non-overlapping pieces, and each piece is further divided into non-overlapping sub-pieces. Specifically, a piece is the unit of video playback and a sub-piece is the unit of data transfer. When a peer enters the system, it first requests the tracker to obtain a set of peers which are called its neighbors. To obtain a particular piece for playback, it seeks download service from its neighboring peers, and each neighboring peer may upload one or more sub-pieces to it. After download all required sub-pieces, this peer can playback them, and it can also upload sub-pieces that it has to its neighbors.

As reported in [18], the distributed nature of P2P streaming systems makes them vulnerable to pollution attack. Specifically, some peers may upload bogus sub-pieces to their neighboring peers, and these peers are called malicious peers or malicious attackers. To guarantee the validity of each piece before playback, a peer must verify all corresponding sub-pieces. However, due to the synchronization requirement and time constraint of P2P streaming systems, one cannot afford to verify every sub-piece, but rather, verify every piece instead. To enable the verification functionality, when the server divides a piece into multiple sub-pieces, it also generates the hash code of that piece and appends it to every associated sub-piece, then uploads all sub-pieces to peers in the system. On the other hand, when a peer wants to playback a piece and has downloaded all required sub-pieces, it first computes the hash value of that piece, then compares it with the hash code appended in each sub-piece. If the hash value matches, the peer can playback that piece. otherwise, it simply drops the whole piece and requests all required sub-pieces from its neighbors again. One thing we want to emphasize is that, by verifying the validity of every piece, even if a peer knows that some sub-pieces are bogus (if the hash value does not match), it still has no clue as to which sub-pieces are bogus and which sub-pieces are valid. In other words, it does not know which neighbors are malicious. The purpose of our detection algorithm is to *identify* the malicious attackers so that one can remove them from the P2P network.

B. Detection Algorithm

According to our general detection framework, we only focus on a particular good (or legitimate) peer, say peer i, and model the interaction between peer i and its neighbors as a repeated process. The detection process proceeds in rounds. We define a round as the time duration of downloading a piece and verifying it. At round t, a subset of neighbors upload sub-pieces to peer i, we call it the uploading set at time t and denote this set as $\mathcal{U}(t)$. According to the definitions in our general detection framework, uploading set is actually the active set at round t, i.e.,

$$\mathcal{A}(t) = \mathcal{U}(t), \qquad t = 1, 2...$$

After downloaded all required sub-pieces, peer i verifies the validity of the piece. If the hash value matches with the appended hash code, then peer i can playback that piece. In other words, the total payoff peer i obtains at round t is $|\mathcal{U}(t)|$ (based on the payoff for each individual interaction as in Equation (1)). On the other hand, if the hash value does not match, then there must be some bogus sub-pieces and peer i cannot playback that piece, so the total payoff that peer i obtains is simply $-\infty$. Formally, we have

$$U(t) = \begin{cases} |\mathcal{U}(t)|, & \text{if the hash value matches,} \\ -\infty, & \text{if the hash value does not match.} \end{cases} \quad (5)$$

Based on the total payoff peer i obtains at round t, the secure set $\mathcal{S}(t)$ can be derived, which is either $\mathcal{U}(t)$ or $\emptyset$ according to the value of the total payoff. Formally, we have

$$\mathcal{S}(t) = \begin{cases} \mathcal{U}(t), & \text{if } U(t) = |\mathcal{U}(t)|, \\ \emptyset, & \text{if } U(t) = -\infty, \end{cases} \qquad t = 1, 2, ... \quad (6)$$

Now, the detection algorithm can be described as follows.

Alg. A1: Defend Against Pollution Attack in P2P Streaming Systems

```
$t \leftarrow 0$;
$\mathcal{M}(0) = \mathcal{N}^i$;
while ($\mathcal{P}_{fp}(t) > \mathcal{P}^*_{fp}$){
    /* shrink the malicious set */
    $t \leftarrow t + 1$;
    $\mathcal{M}(t) \leftarrow \mathcal{M}(t-1) - \mathcal{S}(t)$;
}
remove all peers in $\mathcal{M}(t)$;
```

In this algorithm, the secure set can be derived by Equation (6), while the derivation of the probability of false positive $\mathcal{P}_{fp}(t)$ will be presented in the next subsection. The physical meaning of the termination condition in the while loop is that, if probability of false positive is not smaller than a predefined threshold $\mathcal{P}^*_{fp}$, then we need to continue shrinking the malicious set as some good nodes may reside in it.

C. Performance Analysis

Let us derive the three performance measures to quantify the performance of Alg. A1. We first consider probability of false negative and probability of false positive which quantify the effectiveness of Alg. A1. Results are stated in Theorem 1.

Theorem *1: When Alg. A1 has run for t iterations, probability of false negative $\mathcal{P}_{fn}(t) = 0$ and probability of false positive $\mathcal{P}_{fp}(t) = \prod_{\tau=1}^{t} \frac{\overline{\mathcal{S}(\tau-1)} \cap \overline{\mathcal{S}(\tau)}}{\overline{\mathcal{S}(\tau-1)}}$ where $\overline{\mathcal{S}(\tau)} = \mathcal{N}^i - \mathcal{S}(\tau)$.*

Proof: By using Alg. A1 to identify pollution attackers, note that, only neighbors in the secure set are removed from the malicious set, and the sub-pieces uploaded by neighbors in the secure set are all correct, i.e., the secure set only contains good nodes. Therefore, no malicious peer is wrongly removed from the malicious set, i.e., $\mathcal{P}_{fn}(t) = 0$.

To derive probability of false positive, which is defined as the probability that a good peer is wrongly regarded as a malicious peer, note that, we can formally write it as $\mathcal{P}_{fp}(t) = P\{j \in \mathcal{M}(t)\}$ where j is any good neighbor of peer i. Therefore, we have

$$\begin{aligned} \mathcal{P}_{fp}(t) &= P\{j \in \mathcal{M}(t-1), j \in \mathcal{M}(t)\} \\ &= P\{j \in \mathcal{M}(t\text{–}1)\} P\{j \in \mathcal{M}(t) | j \in \mathcal{M}(t\text{–}1)\} \\ &= \mathcal{P}_{fp}(t-1) P\{j \in \mathcal{M}(t) | j \in \mathcal{M}(t\text{–}1)\}. \end{aligned} \quad (7)$$

To derive $P\{j \in \mathcal{M}(t) | j \in \mathcal{M}(t\text{–}1)\}$ which is the probability that any good node in $\mathcal{M}(t-1)$ is not removed at round t, note that, $\mathcal{M}(t-1) = \mathcal{M}(t-2) \cap \overline{\mathcal{S}(t-1)}$, so on average, the probability that a neighbor of peer i in $\mathcal{M}(t-1)$ is not removed at round t is $\frac{\overline{\mathcal{S}(t-1)} \cap \overline{\mathcal{S}(t)}}{\overline{\mathcal{S}(t-1)}}$. Considering that dishonest peers only account for a small fraction, one can approximate $P\{j \in \mathcal{M}(t) | j \in \mathcal{M}(t\text{–}1)\}$ as $\frac{\overline{\mathcal{S}(t-1)} \cap \overline{\mathcal{S}(t)}}{\overline{\mathcal{S}(t-1)}}$, by substituting it in Equation (7), we have $\mathcal{P}_{fp}(t) = \prod_{\tau=1}^{t} \frac{\overline{\mathcal{S}(\tau-1)} \cap \overline{\mathcal{S}(\tau)}}{\overline{\mathcal{S}(\tau-1)}}$ where $\mathcal{S}(0)$ is initialized as $\emptyset$ and $\mathcal{S}(t)$ is derived by Equation (6). ■

To derive the performance measure of expected number of detection rounds until no good peer exists in the malicious set, we assume that peer i has N neighbors, i.e., $|\mathcal{N}^i| = N$, and k of them are malicious. Moreover, we assume that at each round, each neighbor is selected to upload sub-pieces to peer i with probability α, and this probability is called *uploading probability*. Now, the expected number of detection rounds can be derived and the result is stated in the following theorem.

Theorem *2: If Alg. A1 is used by peer i to identify malicious neighbors, then the expected number of detection rounds is $E[\mathcal{R}] = \sum_{r=1}^{\infty} rP(\mathcal{R} = r)$ where $P(\mathcal{R} = r) = \sum_{d=1}^{r} \binom{r-1}{d-1} (1-\alpha)^{kd} \big(1-(1-\alpha)^k\big)^{r-d} \Big[\big(1-(1-\alpha)^d\big)^{N-k} - \big(1-(1-\alpha)^{d-1}\big)^{N-k}\Big]$.*

Proof: To derive the distribution of $\mathcal{R}$, observe that, the malicious set gets shrunk in a round only when the secure set is not empty, we call such a round a *detectable round* and use r.v. $\mathcal{D}$ to denote the number of detectable rounds peer i needs to identify its malicious neighbors. We have

$$\begin{aligned} P(\mathcal{D} \le d) &= P(\text{after } d \text{ detectable rounds, no good peer} \\ &\qquad \text{exists in the malicious set}) \\ &= \Big(1 - (1-\alpha)^d\Big)^{N-k} \end{aligned}$$

To derive the distribution of $\mathcal{R}$, note that, the probability of a round being detectable is $(1-\alpha)^k$. Given the number of detectable rounds $\mathcal{D}$, the conditional probability $P(\mathcal{R} = r | \mathcal{D} = d)$ is a negative binomial distribution, so we have

$$\begin{aligned} P(\mathcal{R} = r) &= \sum_{d=1}^{r} P(\mathcal{D} = d) P(\mathcal{R} = r | \mathcal{D} = d) \\ &= \sum_{d=1}^{r} \binom{r-1}{d-1} \Big(1-\alpha\Big)^{kd} \Big(1-(1-\alpha)^k\Big)^{r-d} \\ &\qquad \Big[P(\mathcal{D} \le d) - P(\mathcal{D} \le d-1)\Big]. \end{aligned}$$

Now, the expected number of detection rounds $E[\mathcal{R}]$ can be easily derived as shown in the theorem. ∎

IV. Case Study: Wireless Mesh Networks

In this section, we illustrate how to apply our general detection framework on wireless mesh networks in defending against pollution attack. We first provide the background on wireless mesh networks and the related problem of pollution attack, then we propose a fully distributed detection algorithm to identify malicious attackers. Lastly, we quantify the performance of the detection algorithm.

A. Background on Wireless Mesh Networks

Wireless mesh networks usually consist of two types of nodes: mesh routers and mesh clients. Each node operates not only as a host but also as a routing element which forwards packets to other nodes that are not in the direct transmission range of their senders. Although mesh clients can be mobile, mesh routers are usually stationary. For most commonly used architecture of WMNs, there is a backbone network which only consists of mesh routers, and we only focus on such backbone networks in this paper.

Due to the mobility of mesh clients, WMNs can provide easy Internet access. However, because of the spatial and temporal fading of wireless channels, the loss rate in each communication link is usually very high. For example, as reported in [1], half of the operational links have a loss probability greater that 30%. Therefore, traditional routing protocol which determines the next hop in forwarding a packet cannot guarantee high end-to-end throughput. Conversely, opportunistic routing protocol where every node participates in forwarding packets is used to improve the performance. To further improve the spatial reuse, network coding is also employed. By using network coding enabled opportunistic routing protocol, not only the end-to-end throughput is improved, but can also reduce the packet collision so that the network capacity can be further improved. As demonstrated in [13] and systems like COPE [9] and MORE [5], one can achieve the above argument for unicast and multicast data delivery in WMNs. Therefore, network coding enabled opportunistic routing protocol is usually configured in wireless mesh networks, and we focus on such type of WMNs.

Now, let us provide a brief background on network coding and explain why WMNs are vulnerable to pollution attack. The core idea of using network coding is in "packets mixing". Specifically, intermediate nodes along the source-destination path can mix or encode received packets and forward the coded packet to other nodes. Therefore, some nodes may behave maliciously to inject polluted packets into the system and we call these nodes malicious nodes or malicious attackers. As indicated in [7], pollution attack can be easily launched, and some related work, e.g., [8], [10], [12], [20], [21] address the problem of pollution attack, in particular, on detecting the existence of pollution attack and how to discard polluted packets. In here, we present a fully distributed detection algorithm to identify the malicious nodes in WMNs.

B. Detection Algorithm

As mentioned before, in wireless mesh networks with network coding enabled opportunistic routing, intermediate nodes participate in packets mixing and forwarding. Therefore, it is vulnerable to pollution attack in the sense that malicious nodes may inject polluted packets in the system. To guarantee the correctness of forwarded packets, intermediate nodes must perform verification so as to filter out polluted packets. One common approach is to perform hash verification by using homomorphic hash functions [11]. However, because of the high computational cost of modular exponentiation in performing hash verification, verifying every received packet can only achieve at the expense of low system throughput, so it is impractical for WMNs [7]. To design practical verification schemes, batch verification in which multiple packets are verified at one time are employed. The rational for batch verification is that a set of coded packets can be mixed to a single coded packet by using a set of random linear coefficient, and the mixed coded packet can be verified by using the homomorphic hash function. To verify a set of packets by using batch verification, if the verification passes, i.e., the mixed packet is correct, then it means that the set of packets are all correct. On the other hand, if the verification does not pass, i.e., the mixed packet is not correct, there must be some polluted packets. However, the intermediate node does not know which packets are polluted and it can not identify the nodes which forward polluted packets to it either. In here, we present a fully distributed detection algorithm to identify the malicious nodes which inject polluted packets.

Based on our general detection framework, we only focus on a particular node, say node i, and call the set of nodes which are in the direct transmission range of node i the neighboring nodes and also use $\mathcal{N}^i$ to denote it. As an intermediate node, node i receives packets from its neighbors, performs batch verification to verify received packets and forwards the encoded packet by mixing received packets. We model the interaction of node i and its neighbors as a repeated process and time proceeds in rounds. We define one round as the time duration of performing one batch verification and the time of receiving the packets that are verified by the batch verification. At round t, node i receives a set of packets from its neighbors, and we call the set of neighbors which forward packets to it *the forwarder set*, and use $\mathcal{F}(t)$ to denote it. Actually, the forwarder set is the active set we defined in our general detection framework, we have

$$\mathcal{A}(t) = \mathcal{F}(t), \quad t = 1, 2...$$

After receive multiple packets, node i needs to perform batch verification to verify them, i.e., to verify the mixture of the received packets. If the mixed packet is correct, then node i can forward it to other nodes. Therefore, the total payoff node i obtains in this round is $|\mathcal{F}(t)|$. On the other hand, if the mixed packet is not correct, then some of the received packets must be polluted and node i has no idea which packets are polluted but can only drop them, so the total payoff it obtains is $-\infty$. Formally, we have

$$U(t) = \begin{cases} |\mathcal{F}(t)|, & \text{if the batch verification passes,} \\ -\infty, & \text{if the batch verification does not pass.} \end{cases}$$

Based on the total payoff node i obtains at round t, the secure set $\mathcal{S}(t)$ can be derived which is either $\mathcal{F}(t)$ or $\emptyset$ according to the value of the total payoff. Formally, we have

$$\mathcal{S}(t) = \begin{cases} \mathcal{F}(t), & \text{if } U(t) = |\mathcal{F}(t)|, \\ \emptyset, & \text{if } U(t) = -\infty, \end{cases} \quad t = 1, 2, ... \tag{8}$$

Now, the detection algorithm can be described as follows.

Alg. A2: Defend Against Pollution Attack in WMNs

```
$t \leftarrow 0$;
$\mathcal{M}(0) = \mathcal{N}^i$;
while ($\mathcal{P}_{fp}(t) > \mathcal{P}^*_{fp}$){
    /* shrink the malicious set */
    $t \leftarrow t + 1$;
    $\mathcal{M}(t) \leftarrow \mathcal{M}(t-1) - \mathcal{S}(t)$;
}
remove all nodes in $\mathcal{M}(t)$;
```

In this algorithm, the secure set can be derived by Equation (8) and the probability of false positive $\mathcal{P}_{fp}(t)$ will be derived in next subsection. Again, the physical meaning of the termination condition in the while loop is that if the probability of false positive is not small enough, e.g., it is not smaller than a predefined threshold $\mathcal{P}^*_{fp}$, then we need to continue shrinking the malicious set as some good nodes still remain in it.

C. Performance Analysis

In here, we derive the three measures, probability of false negative, probability of false positive and expected number of detection rounds to quantify the performance of Alg. A2.

We assume that node i has N neighbors, i.e., $|\mathcal{N}^i| = N$, and k of them are malicious nodes which inject polluted packets into the system. Since fairness is a built-in feature in medium access control (MAC) protocol in wireless networks, a node cannot monopolize the channel to forward packets. Specifically, 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. We define α as the probability that a neighbor of node i performs forwarding at each round, which is called *forwarding probability*. Note that, in WMNs, when a source disseminates a file to destinations, it first breaks up the file into multiple generations, and the source moves to another generation only when the destinations have received all packets of one generation. Usually, a generation contains 32 independent packets and a node may perform multiple verifications during the period of transmitting one generation. Last but not least, when a node is ready to transmit, it may perform multiple transmissions. Therefore, α is usually less than one so that the neighbors of node i can be distinguished and the malicious set can get shrunk.

Now, we can derive the performance measures of Alg. A2 and the results are stated in Theorem 3. Since the proof is very similar to the case of P2P networks, we omit it here.

Theorem *3: When Alg. A2 has run for t iterations, probability of false negative $\mathcal{P}_{fn}(t) = 0$ and probability of false positive $\mathcal{P}_{fp}(t) = \prod_{\tau=1}^{t} \frac{\overline{\mathcal{S}(\tau-1)} \cap \overline{\mathcal{S}(\tau)}}{\overline{\mathcal{S}(\tau-1)}}$ where $\overline{\mathcal{S}(\tau)} = \mathcal{N}^i - \mathcal{S}(\tau)$ and $\mathcal{S}(\tau)$ is derived by Equation (8). The expected number of rounds needed for detection until no good node exists in the malicious set is $E[\mathcal{R}] = \sum_{r=1}^{\infty} rP(\mathcal{R} = r)$ where $P(\mathcal{R} = r) = \sum_{d=1}^{r} \binom{r-1}{d-1} (1-\alpha)^{kd} \big(1-(1-\alpha)^k\big)^{r-d} \Big[\big(1-(1-\alpha)^d\big)^{N-k} - \big(1-(1-\alpha)^{d-1}\big)^{N-k}\Big]$.*

V. Case Study: Online Social Networks

Let us apply the general detection framework to OSNs to identify dishonest recommenders who give wrong recommendations. We first provide a brief background on OSNs and the shill attack problem, then present a distributed detection algorithm to identify the dishonest recommenders.

A. Background on OSNs and Shill Attack

Online social networks such as Facebook and Twitter have attracted a lot of users. Moreover, many users have integrated these sites into their daily activities, e.g., they interact with their friends very frequently and they often seek recommendations from their friends before they do purchases. On the other hand, when one buys a product, she may also make recommendations on this product to her friends so that her friends' decisions are influenced. Also, one may forward received recommendations to other people, which may also influence other people's decisions. Such influence caused by recommendations are called word-of-mouth effect, and it makes the purchase behavior spread quickly in an OSN. This way of advertisement which is based on the word-of-mouth effect is called the viral marketing, and it is very effective in increasing revenue for a company.

However, viral marketing also opens door for potential security attack as some people may make wrong recommendations to their friends so as to mislead their purchases. We call such an attack *the shill attack*. This attack may be launched by users in an OSN who are hired by companies to promote their products, or worse yet, provide wrong recommendations to badmouth their competitors' products. If shill attack is launched in an OSN, due to the misleading recommendations made by the attackers, a low quality product may still be purchased but products which have high intrinsic quality may lose out. We call the users in an OSN who deliberately make wrong recommendations the dishonest recommenders or the attackers. Note that, even if a user is honest, she may also give wrong recommendations to her friends as she may simply forward received recommendations which are made by her dishonest friends. Therefore, one cannot determine whether her friend is honest or not simply based on the recommendation. The aim of the next subsection is to present a detection algorithm to identify dishonest recommenders in OSNs so as to defend against shill attack.

B. Detection Algorithm

In this subsection, we develop a fully distributed detection algorithm to identify dishonest recommenders in OSNs. We focus on a particular user, say user i, and use $\mathcal{N}^i$ to denote the set of friends of user i. We consider a set of substitutable products P_1, P_2, ..., P_M, which are produced by firms F_1, F_2, ..., F_M respectively, and these firms compete in the same market. Two products are substitutable if they are compatible, e.g., polo shirts from brand X and brand Y are substitutable goods from the customers' points of view. We assume that one can estimate the quality of a product if she buys it, and let q be the evaluation function which reveals the intrinsic quality of a product. Moreover, we assume that each product is either of high quality or of low quality, formally, we have

$$q(P_j) = \begin{cases} 1, & \text{if product } P_j \text{ is of high quality,} \\ 0, & \text{if product } P_j \text{ is of low quality.} \end{cases}$$

We model the purchase experience of user i as a repeated process and time proceeds in rounds. We define one round as the time duration of purchasing one product. At round t, user i purchases a product, say P_j, and she may also receive some recommendations on product P_j from her friends. Some recommendations may give high rating on P_j and others may give low rating on P_j. Formally, we define a recommendation which gives high rating on P_j as *positive recommendation* and use $\mathcal{R}_P(P_j)$ to denote it. Correspondingly, a recommendation which gives low rating on P_j is defined as *negative recommendation* and we use $\mathcal{R}_N(P_j)$ to denote it. We have

$$\begin{cases} \mathcal{R}_P(P_j) = H(P_j), \\ \mathcal{R}_N(P_j) = L(P_j), \end{cases}$$

where $H(P_j)$ means giving high rating on product P_j and $L(P_j)$ means giving low rating on P_j.

To take the intrinsic quality of a product into consideration, we have the definitions of correct recommendation and wrong recommendation. Formally, if a recommendation gives high rating on product P_j if it is of high quality and gives low rating on P_j if it is of low quality, then it is called a *correct recommendation*. Correspondingly, a *wrong recommendation* gives high rating on P_j if P_j is of low quality and gives low rating on P_j if P_j is of high quality. Formally, we have

$$\begin{aligned} \mathcal{R}_C(P_j) &= \begin{cases} H(P_j), & \text{if } q(P_j) = 1, \\ L(P_j), & \text{if } q(P_j) = 0, \end{cases} \\ \mathcal{R}_W(P_j) &= \begin{cases} H(P_j), & \text{if } q(P_j) = 0, \\ L(P_j), & \text{if } q(P_j) = 1. \end{cases} \end{aligned}$$

Since user i buys product P_j at round t and she may also receive some recommendations from her friends. To distinguish her friends, we define $\mathcal{N}_C(t)$ as the set of friends who give her correct recommendations. Correspondingly, $\mathcal{N}_W(t)$ and $\mathcal{N}_N(t)$ are defined as the set of friends who give her wrong recommendations and no recommendation respectively. Obviously, we have $\mathcal{N}^i = \mathcal{N}_C(t) \cup \mathcal{N}_W(t) \cup \mathcal{N}_N(t)$.

For each user, if she is honest, then she will give correct recommendations to her friends as long as she knows the intrinsic quality of a product (e.g., she buys the product). On the other hand, even if she does not know the intrinsic quality of a product, she may still give recommendations based on received recommendations from her friends. Hence, she may make wrong recommendations in this case as she may be misled by her friends. Therefore, a friend who gives wrong recommendation is not definitely dishonest, but rather *potentially dishonest*. For dishonest users, usually their goal is to promote a product of low quality and at the same time, badmouth all other products, so the intuitive strategy of dishonest users is to give positive recommendations on the product they aim to promote and give negative recommendations on other products. Note that, dishonest recommenders may have different types, i.e., they may try to promote different products. For example, users who are hired by firm F_i try to promote product P_i, but users who are hired by firm F_j try to promote P_j. Without loss of generality, we assume that there are m types of dishonest recommenders who are hired by firms F_1, F_2, ..., F_m and try to promote P_1, P_2, ..., P_m respectively. We assume that the products promoted by dishonest recommenders are all of low quality. Therefore, the strategy of dishonest users of type l $(1 \leq l \leq m)$ can be formally described as follows.

$$\mathcal{R}_P(P_l) \wedge [\wedge_{j=1,j\neq l}^{M} \mathcal{R}_N(P_j)]. \tag{9}$$

To apply our general detection framework, we need to first determine the secure set at each round. Since we only focus on a particular honest user i. At round t, we assume that user i buys product P_j. If P_j is of low quality, then user i may be influenced by the wrong recommendations received from her friends when she makes the decision. So the total payoff user i obtains at this round must be $-\infty$. On the other hand, if P_j is of high quality, then it means that user i is not affected by wrong recommendations, so we can ignore the wrong recommendations and only take the set of friends who give correct recommendations as the active set, i.e., $\mathcal{A}(t) = \mathcal{N}_C(t)$. The total payoff user i obtains at this round is $|\mathcal{N}_C(t)|$. Therefore, the secure set is the same as the active set. Formally, the secure set can be stated as follows.

$$\mathcal{S}(t) = \begin{cases} \mathcal{N}_C(t), & \text{if } q(P_j) = 1, \\ \emptyset, & \text{if } q(P_j) = 0, \end{cases} \quad t = 1, 2, ... \tag{10}$$

where P_j is the product that user i buys at round t.

Now, the detection algorithm can be described as follows.

Alg. A3: Defend Against Shill Attack in OSNs

```
$t \leftarrow 0$;
$\mathcal{M}(0) = \mathcal{N}^i$;
while ($\mathcal{P}_{fp}(t) > \mathcal{P}_{fp}^*$){
    /* shrink the malicious set */
    $t \leftarrow t + 1$;
    $\mathcal{M}(t) \leftarrow \mathcal{M}(t-1) - \mathcal{S}(t)$;
}
remove all users in $\mathcal{M}(t)$;
```

C. Performance Analysis

Let us derive the three performance measures to quantify the performance of Alg. A3. We first consider probability of false negative and probability of false positive. The results are stated in Theorem 4 and we omit the proof since its derivation is similar with the previous applications.

Theorem 4: *When Alg. A3 has run for t iterations, probability of false negative $\mathcal{P}_{fn}(t) = 0$ and probability of false positive $\mathcal{P}_{fp}(t) = \prod_{\tau=1}^{t} \frac{\overline{\mathcal{S}(\tau-1)} \cap \overline{\mathcal{S}(\tau)}}{\overline{\mathcal{S}(\tau-1)}}$ where $\overline{\mathcal{S}(\tau)} = \mathcal{N}^i - \mathcal{S}(\tau)$ and $\mathcal{S}(\tau)$ is derived by Equation (10).*

To derive the expected number of detection rounds until no honest user exists in the malicious set, note that, an honest friend will be removed from the malicious set at some round only when two conditions are satisfied. Firstly, user i buys a product of high quality at that round. We call such a round *a detectable round* and use p_d to denote the probability that a round is detectable. Secondly, the honest friend must give correct recommendations to user i on the product user i purchases. In the following, we first derive this probability, i.e., the probability that an honest friend gives correct recommendations at each round, and we use p_{hc} to denote it. We assume that, when an honest user does not know the intrinsic quality of a product and gives recommendations based on received recommendations, she adopts the majority rule [3], [19]. Specifically, for an honest user, if more than half of her friends give her positive (negative) recommendations, she also gives positive (negative) recommendations to others. Otherwise, she gives no recommendation. Therefore, an honest person gives correct recommendations if and only if she buys a product or more than half of her friends give her correct recommendations. By employing the local mean field technique proposed in [17], [22], we can derive p_{hc} as stated in the following lemma.

Lemma 1: *If honest users adopt majority rule to provide recommendations when they do not know the intrinsic quality of a product, then p_{hc} can be derived as follows.*

$$1-E[Y] = (1-\mu)\sum_{k=0}^{\infty}\sum_{j=0}^{\lfloor\frac{1}{2}(k+1)\rfloor} P_1(k+1)C_k^j E[Y]^j(1-E[Y])^{k-j},$$

$$1-p_{hc} = (1-\mu)\sum_{k=1}^{\infty}\sum_{j=0}^{\lfloor\frac{1}{2}k\rfloor} P_0(k)C_k^j E[Y]^j(1-E[Y])^{k-j}.$$

where μ is the market share of the product, $P_0(k)$ is the degree distribution of the social network and $P_1(k)$ is the degree distribution of descendant nodes in the social network.

Now, given the probability of a round being detectable, i.e., p_d, and probability of an honest user giving correct recommendations, i.e., p_{hc}, we can derive the expected number of detection rounds as stated in the following theorem.

Theorem 5: *When Alg. A3 is used by user i to identify dishonest friends, the expected number of detection rounds is $E[\mathcal{R}] = \sum_{r=1}^{\infty} rP(\mathcal{R} = r)$ where $P(\mathcal{R} = r) = \sum_{d=1}^{r} \binom{r-1}{d-1} p_d^d(1-p_d)^{r-d}\Big[\big(1-(1-p_{hc})^d\big)^{N-k} - \big(1-(1-p_{hc})^{d-1}\big)^{N-k}\Big]$, where N is the number of friends of user i and k of them are dishonest.*

Proof: Similar with the proof of Theorem 2, we can first derive the distribution of number of detectable rounds needed for detection and we also use r.v. $\mathcal{D}$ to denote it, we have

$$\begin{aligned} P(\mathcal{D} \le d) = &P\{\text{after } d \text{ detectable rounds, no honest} \\ &\text{friend exists in the malicious set}\} \\ = &\big(1-(1-p_{hc})^d\big)^{N-k}. \end{aligned}$$

Again, 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

$$\begin{aligned} P(\mathcal{R} = r) &= \sum_{d=1}^{r} P(\mathcal{D} = d)P(\mathcal{R} = r|\mathcal{D} = d) \\ &= \sum_{d=1}^{r} \binom{r-1}{d-1} p_d^d(1-p_d)^{r-d} \\ &\quad \Big[P(\mathcal{D} \le d) - P(\mathcal{D} \le d-1)\Big], \end{aligned}$$

where p_d is the probability of a round being detectable, i.e., probability that the product user i buys is of high quality, and it can be estimated based on the purchase history of user i. Now, the expected number of detection rounds, $E[\mathcal{R}]$ can be easily derived as shown in Theorem 5. ■

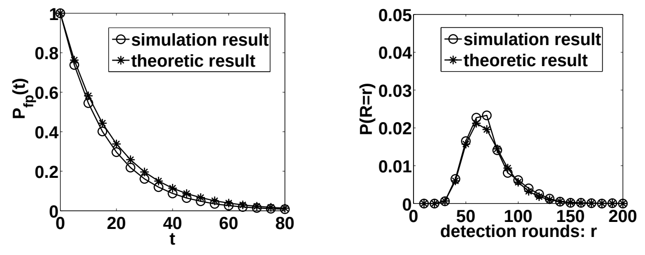
VI. Simulation and Model Validation

In this section, we validate our analysis via simulation so as to further show the effectiveness and efficiency of our detection algorithms. Since the analysis on Alg. A1 and Alg. A2 are almost the same and due to page limit, we only present the results on P2P streaming systems and OSNs.

A. P2P Streaming Systems

In this subsection, we validate our analysis on defending against pollution attack in P2P streaming systems by using Alg. A1. In particular, we consider probability of false positive and the distribution of number of detection rounds.

In this simulation, we assume that peer i has 50 neighbors, i.e., $\mathcal{N}^i = 50$, and five of them are malicious attackers, i.e., $k = 5$. Moreover, at each round, each neighbor of peer i is chosen to upload with probability 0.1, i.e., the uploading probability α is set as 0.1. Figure 1a shows the results of probability of false positive. In this figure, the horizontal axis is the number of rounds or iterations that Alg. A1 has been executed, and the vertical axis shows the probability of false positive. We have two curves in this figure. The curve with stars shows the theoretic result which is derived in Theorem 1 and the curve with circles shows the simulation result which is derived based on the definition, precisely, $\frac{|\mathcal{M}(t)|-k}{N-k}$. We can see that probability of false positive converges to 0 eventually, which means that malicious set only contains malicious attackers as long as detection time is long enough. Moreover, probability of false positive decreases to a reasonably small value, say 0.1, even if the algorithm has only run a small number of rounds, say 50 rounds, which indicates the rationale of the termination condition in our detection algorithm Alg. A1.



(a) Probability of false positive. (b) Number of detection rounds.

Fig. 1: Performance of Alg. A1

The result on the distribution of number of detection rounds is shown in Figure 1b. In this figure, the horizontal axis represents the number of rounds needed to detect until no good peer exists in malicious set, and the vertical axis shows the probability. The curve with stars shows the theoretic result which is derived in Theorem 2, and the corresponding expectation is $E[\mathcal{R}] = 71$. The curve with circles shows the simulation result, which is obtained by running our detection algorithm 10000 times and in each time, we record the number of rounds needed for detection. The corresponding expectation is $E[\mathcal{R}] = 72$. To make the figure be clear, we only show probabilities when the number of detection rounds is times of ten. We can see that most of the time, one only needs less than 100 rounds to detect its malicious neighbors even if it has fifty neighbors and five of them are malicious. In summary, the results shown in Figure 1 not only validate our analysis on the performance of Alg. A1, but also show the effectiveness and efficiency of our detection algorithm.

B. Online Social Networks

In this subsection, we validate the analysis of Alg. A3 which defends against shill attack in OSNs. Note that an honest friend of user i giving correct recommendations does not only depend on her own purchasing behaviors, but also depends on her friends' recommendations, i.e., user i may also be influenced by her friends' friends, or even a user which is far away from user i as wrong recommendations may be spread through the network. We can not model the underlying network as a star network. Conversely, we synthesize a dynamically evolving social network which can simulate the behaviors of users. We then examine the impact of shill attack and validate the performance analysis of our detection algorithm.

Synthetic dynamically evolving OSNs: To synthesize a dynamic OSN to simulate the behaviors of users, we need to make assumptions on (1) how people make recommendations to their friends, (2) how people make decisions on purchasing which product, and (3) how fast the recommendations spread.

Firstly, dishonest users follow the strategy defined in Formula (9). For an honest user, if she knows the intrinsic value of a product, she gives correct recommendations on that product, otherwise, she gives recommendations by using majority rule. Specifically, she gives positive (or negative) recommendations to others if and only if more than half of her friends give her positive (or negative) recommendations.

Secondly, when an honest user decides to buy a product, she buys the product with maximum number of effective recommendations. The number of effective recommendations is defined as the difference between number of positive recommendations and number of negative recommendations. The rational is that one buys a product which receives high rating as many as possible and low rating as few as possible.

Lastly, we assume that recommendations broadcast much faster than users' purchasing rate, Precisely, after a user gives a recommendation to her friends, her friends update the number of received recommendations, if majority rule is satisfied, then they further make recommendations to their friends, and this process continues until no one in the system can make a recommendation, and the whole broadcasting process finishes before the next purchase instance in the whole system.

In this simulation, we employ the GLP model [4] which is based on preferential attachment [2] to generate a scale-free graph with power law degree distribution and high clustering coefficient. The generated graph has around 8,000 nodes, 70,000 edges and clustering coefficient of around 0.3. We assume that there are five products, $P_1, ..., P_5$ and system starts from the "uniform" state in which all products have the same market share. We randomly choose an honest user as the detector, and during one detection round, $10\%|V|$ purchase instances happen where $|V|$ is the total number users in the system. One thing we want to emphasize is that the assumptions in this section are only for the simulation purpose.

Impact of shill attack: To explore the impact of shill attack, we let the system start from "0" state where no purchase happens and evolves until 10,000 purchases are made. We assume that P_1 is of low quality and all other products are of high quality. Our objective is to count what is the fraction of purchases for each product out of the total 10,000 purchases. We run the simulation multiple times to take the average value, and the result is shown in Figure 2. We can see that if no dishonest user exits in the network, product P_1 is only purchased with a very small probability. The reason why the probability is not zero is that if a user receives no recommendation, then she just randomly purchases a product. However, if we set 5% of users as dishonest recommenders and let them promote product P_1, then P_1 is purchased with probability greater than 0.15. In a summary, shill attack can distort the normal sales distribution severely.

Analysis validation: In this simulation, we randomly set 0.1% of users as dishonest recommenders and randomly choose an honest user who has dishonest friends as the detector. We first consider probability of false positive, and the result is shown in Figure 3a. We can see that after only a small number of rounds (< 10), probability of false positive quickly converges to 0, which means that the detected users are really dishonest. In other words, our algorithm is effective in identifying dishonest recommenders.

The result of the distribution of number of detection rounds

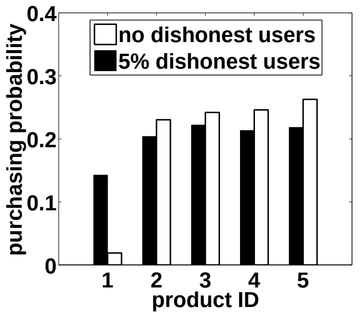
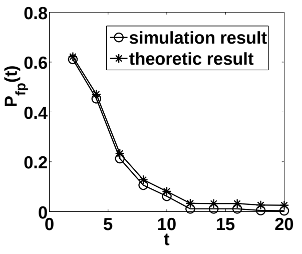
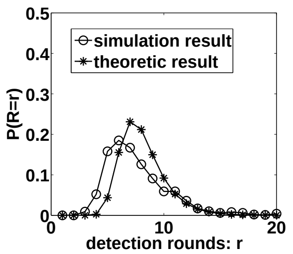


Fig. 2: Impact of shill attack.



(a) Probability of false positive.



(b) Number of detection rounds.

Fig. 3: Performance of Alg. A3

is shown in Figure 3b. The curve with stars shows the theoretic result which is derived in Theorem 5, and the curve with circles shows the simulation result which is obtained by running our detection algorithm multiple times, and in each time, we record the number of rounds needed for detection. We can see that the theoretic result does not fit with the simulation result very well. The reason is that, the probability of a round being detectable is estimated based on detection history. Besides, the probability that an honest user gives correct recommendations which is derived by using local mean field is an average value over all users, but in the simulation, the simulation result is for a particular user which is randomly chosen from the network. However, if we look at the expectation, the theoretic result is very close to the simulation result. Last but not least, our algorithm still works well even without deriving the performance measure of expected number of rounds, and this performance measure is only used for estimating the efficiency of our detection algorithm.

VII. Conclusion

Due to the intrinsic property that nodes need to interact with each other, interactive networks are vulnerable to various attacks which are launched by malicious nodes. In this work, we develop a general and fully distributed detection framework to identify malicious nodes in interactive networks. We illustrate this framework on P2P networks, WMNs and OSNs, and present three algorithms which can defend against pollution attacks in P2P streaming systems and WMNs, as well as shill attacks in OSNs. Our algorithms allow each node independently perform the detection to identify its malicious neighbors. Mathematical analysis is provided to quantify the effectiveness and efficiency of our detection algorithms. We also validate our models via extensive simulations. Our detection framework can be viewed as an indispensable tool to maintain the viability of interactive networks.

Acknowledgement: this research is supported in part by the Institute of Network Coding in CUHK and RGC grant 415310.

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] A.-L. Barabasi and R. Albert. Emergence of Scaling in Random Networks. *Science*, 1999.

[3] E. Berger. Dynamic Monopolies of Constant Size. *J. Comb. Theory Ser. B*, 83(2):191–200, 2001.

[4] T. Bu and D. Towsley. On Distinguishing between Internet Power Law Topology Generators. *INFOCOM*, 2002.

[5] S. Chachulski, M. Jennings, S. Katti, and D. Katabi. Trading Structure for Randomness in Wireless Opportunistic Routing. In *SIGCOMM '07*, pages 169–180, New York, NY, USA, 2007.

[6] P. Dhungel, X. Hei, K. W. Ross, and N. Saxena. "The Pollution Attack in P2P Live Video Streaming: Measurement Results and Defenses". P2P-TV '07, 2007.

[7] 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*, pages 111–122, 2009.

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] Y. K. Li and J. C. S. Lui. Stochastic Analysis of A Randomized Detection Algorithm for Pollution Attack in P2P Live Streaming Systems. *IFIP Performance Conference*, 2010.

[15] Y. K. Li and J. C. S. Lui. Friends or Foes: Detecting Dishonest Recommenders in Online Social Networks. *IEEE 20th Int. Conference on Computer Communciation Networks (ICCCN)*, 2011.

[16] Y. K. Li and J. C. S. Lui. Identifying Pollution Attackers in Network-Coding Enabled Wireless Mesh Networks. *IEEE 20th Int. Conference on Computer Communciation Networks (ICCCN)*, 2011.

[17] Y. K. Li, B. Q. Zhao, and J. C. S. Lui. On Modeling Product Advertisement in Large Scale Online Social Networks. *Accepted for publication in the IEEE/ACM Transactions on Networking.*

[18] J. Liang, R. Kumar, Y. Xi, and K. Ross. "Pollution in P2P File Sharing Systems". INFOCOM, March 2005.

[19] D. Peleg. Local Majority Voting, Small Coalitions and Controlling Monopolies in Graphs: A Review. Technical report, Israel, 1996.

[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*, pages 788–792, Piscataway, NJ, USA, 2009. IEEE Press.

[22] B. Q. Zhao, Y. K. Li, J. C. S. Lui, and D. M. Chiu. Mathematical Modeling of Advertisement and Influence Spread in Social Networks. *ACM Workshop on the Economics of Networked Systems (NetEcon)*, 2009.