

# Detecting Colluding Blackhole and Greyhole Attacks in Delay Tolerant Networks

Thi Ngoc Diep Pham and Chai Kiat Yeo

**Abstract**—Delay Tolerant Network (DTN) is developed to cope with intermittent connectivity and long delay in wireless networks. Due to the limited connectivity, DTN is vulnerable to blackhole and greyhole attacks in which malicious nodes intentionally drop all or part of the received messages. Although existing proposals could accurately detect the attack launched by individuals, they fail to tackle the case that malicious nodes cooperate with each other to cheat the defense system. In this paper, we suggest a scheme called Statistical-based Detection of Blackhole and Greyhole attackers (SDBG) to address both individual and collusion attacks. Nodes are required to exchange their encounter record histories, based on which other nodes can evaluate their forwarding behaviors. To detect the individual misbehavior, we define forwarding ratio metrics that can distinguish the behaviour of attackers from normal nodes. Malicious nodes might avoid being detected by colluding to manipulate their forwarding ratio metrics. To continuously drop messages and promote the metrics at the same time, attackers need to create fake encounter records frequently and with high forged numbers of sent messages. We exploit the abnormal pattern of appearance frequency and number of sent messages in fake encounters to design a robust algorithm to detect colluding attackers. Extensive simulation shows that our solution can work with various dropping probabilities and different number of attackers per collusion at high accuracy and low false positive.

**Index Terms**—Blackhole attack, greyhole attack, flooding attack, collusion, encounter record, DTN

## 1 INTRODUCTION

DELAY Tolerant Network (DTN) [1], [2], [3] has been developed as the solution to challenging networks characterized by intermittent connectivity, long delay or frequent disruption. DTN makes use of hop-by-hop routing and the store-and-forward paradigm to overcome the lack of end-to-end paths. DTN has been widely embraced for many practical applications such as interplanetary networks [4], sensor networks in extreme environment [5], [6], low-cost Internet connection to isolated areas [7] and high-speed vehicular network [8].

DTN is threatened by various attacks, including blackhole and greyhole. Blackhole attackers drop all the received messages even if they have enough buffer storage. Greyhole attackers drop a fraction of received messages to avoid arousing suspicion and detection from other nodes. The dropping misbehavior will decrease the overall message delivery and waste the resources of intermediate nodes that have carried and forwarded the dropped messages. The lack of continuous path and limited connectivity resources in DTN make the detection of these attacks more challenging than that in a well-connected ad hoc network.

Blackhole and greyhole attacks in DTN have been studied in [9], [10], [11], [12], [13], [14], [15]. Chuah et al. [9], [10] require trusted ferries to check if nodes arbitrarily increase their delivery probabilities to absorb more data, which is preliminary to the dropping attack. Gao et al. [11] depends on a trusted authority to investigate nodes based on the

evidences of forwarding tasks, contact opportunities and actual forwarding behaviors. Li and Das [12] designs a trust-based framework in which the message forwarding behavior is acknowledged and rewarded with reputation. Li et al. [13], Guo et al. [14], and Li and Cao [15] make use of nodes' encounter records to detect or mitigate the impact of the attack.

While the schemes mentioned above can detect individual attackers well, they cannot handle the case where attackers cooperate to avoid the detection. For example, in an individual detection scheme, nodes are evaluated by their histories of encounters with other nodes, i.e., encounter records. A node is judged as malicious if its forwarding ratio, which is the ratio between the total number of sent and received messages, is lower than a threshold. This works since attackers tend to drop received messages instead of sending them out. However, if attackers cooperate to boost the forwarding ratio by creating forged encounter records for each other, they can avoid being detected by the scheme. We call this attack as collusion attack. As shown in Fig. 1a,  $n_3$  can detect  $m$  as malicious when judging its authentic records with  $n_1$ ,  $n_2$ . However, Fig. 1b shows that by colluding with  $m'$ ,  $m$  can produce fake records to trick  $n_3$  to judge it as normal.

Unlike most of the previous works, in this paper, we propose a scheme, Statistical-based Detection of Blackhole and Greyhole (SDBG), which can detect both individual and collusion attacks with high accuracy. SDBG is designed based on the following observations. To detect individual attack, the forwarding ratio defined above can be used. Moreover, attackers tend to send out their own messages rather than messages of other nodes. We can thus enhance the individual detection accuracy by further observing the number of self-sent messages. In collusion attack, since attackers need to frequently create fake encounter records to boost the forwarding

• The authors are with the School of Computer Engineering, Nanyang Technological University, Block N4 Nanyang Avenue #02a-32, Singapore 639798. E-mail: pham0069@e.ntu.edu.sg, asckyeo@ntu.edu.sg.

Manuscript received 16 Nov. 2014; revised 19 June 2015; accepted 5 July 2015. Date of publication 20 July 2015; date of current version 31 Mar. 2016. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2015.2456895

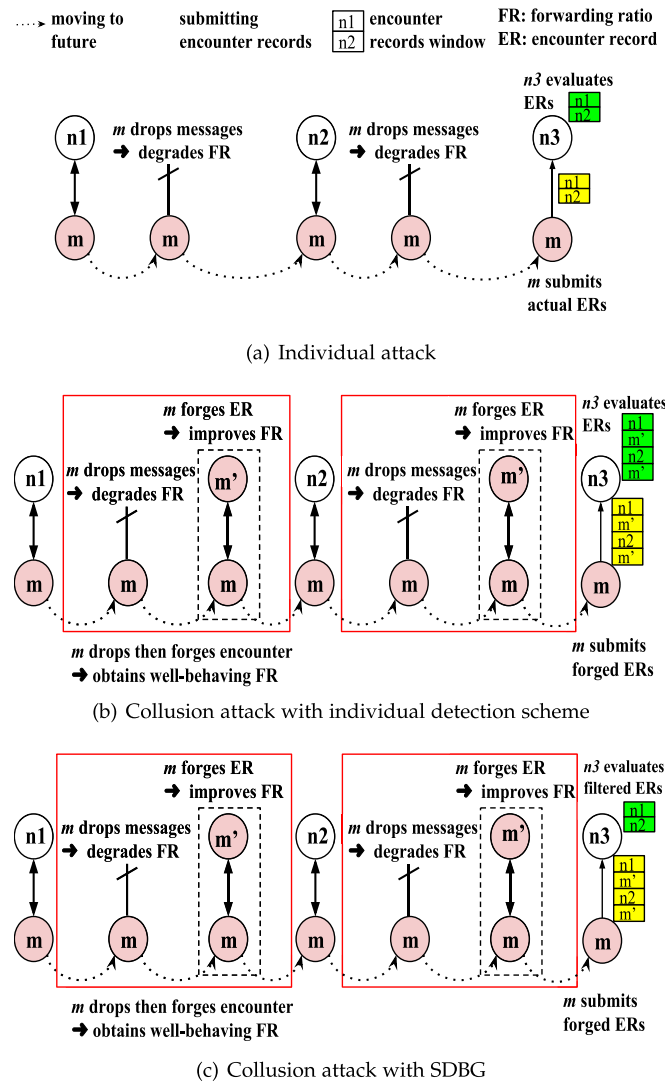


Fig. 1. Examples of attacks and detection schemes.  $n_1, n_2, n_3$  are normal nodes.  $m$  and  $m'$  are malicious nodes.

ratio metric, their number of sent messages can be high. This observation enables us to detect collusion attack.

We illustrate in more details how SDBG detects collusion attack in Fig. 1c. To persistently avoid the detection,  $m$  and its colluder  $m'$  have to create a large number of forged records. To compensate for dropping messages received from  $n_1$  and  $n_2$ , these fake records have to include high number of sent messages. When it encounters  $n_3$ ,  $m$  submits the forged record window which includes contacts with  $n_1$ ,  $m'$ ,  $n_2$  and  $m'$ . By observing the high encounter frequency and number of sent messages between  $m$  and  $m'$ ,  $n_3$  suspects  $m'$  as colluder of  $m$ . To confirm that,  $n_3$  can exclude the suspiciously fake records between  $m$  and  $m'$  from the submitted history of  $m$ . In this way, it will obtain  $m$ 's actual encounter history which includes contacts with  $n_1, n_2$  only. The real forwarding ratios inferred from the authentic window allow  $n_3$  to detect  $m$  as malicious.

Motivated by those observations, we design a detection scheme which involves two processes: individual and collusion detection. Upon encounter, nodes exchange encounter records and use them to judge each other as malicious or not. First, the judging node implements the individual

detection algorithm. It computes the forwarding metrics of the judged node and compares them against predefined thresholds. If the thresholds are violated, the judging node decreases the trust in the judged one. Otherwise, the judging node continues with the process of collusion detection which involves two phases: suspicion and confirmation. The first phase lists the suspicious colluders with the judged node. Nodes are suspected if the total number of messages they receive from the judged node is abnormally high. However, normal nodes, who have close relationship with the judged node, may also encounter and exchange messages a lot with it. To avoid the misjudgment, the second phase is introduced to verify the suspicion. In this phase, the judging node omits the encounter records between suspicious colluders and only uses the remaining records to assess the judged node's forwarding ratios.

To validate our scheme, we implement extensive simulations in ONE simulator. Simulation results show that SDBG works for a wide range of dropping probabilities in various settings of colluding attackers and different routing protocols. SDBG achieves a detection rate of at least 70 percent with false positive rate below 1.6 percent. Compared to state-of-the-art schemes [14], [15], SDBG can detect collusion attack with higher accuracy and lower false positive rate.

The rest of the paper is organized as follows. Section 2 describes the system model. Sections 3 and 4 explain the detection approach of individual and collusion attack respectively. The simulation results and evaluations are presented in Section 5. Section 6 summarizes the existing works against blackhole and greyhole attacks in DTN. Section 7 concludes the paper.

## 2 SYSTEM MODEL

In this section, we introduce the system setting under DTN scenario and describe the behavior model of normal nodes and individual attackers.

### 2.1 Mobile Node Model

We consider the DTN mobile network scenario which involves mobile nodes communicating in ad-hoc mode with wireless technology such as Wi-fi. We assume a trusted authority with the right to assign each node a unique identifier and a pair of public and private keys. Nodes are assumed to know the public keys of each other so that they can authenticate messages signed by others.

Under the above settings, we model the general behaviors of nodes as follows. When two nodes encounter and exchange messages, each of them generates an Encounter Record (ER) and stores it in its own storage. The ER includes the identities of two nodes, the ER sequence numbers assigned by them, the encounter timestamp and the lists of sent and received messages between the two parties and their signatures.

To ensure that nodes agree on the encounter timestamp in ER, the network is loosely time synchronized. This means that the time is divided into slots of equal duration. At any one time, any two nodes have their local clock times falling in the same time slot. Given that in DTN which is often characterized by sparse nodes and infrequent encounters, the inter-contact time is usually in minutes if not hours. We

can set the time slot to any length that is suitable for the DTN concerned. The time slot can therefore be in duration of minutes.

Suppose the two nodes  $i$  and  $j$  with respective identifiers  $ID_i$  and  $ID_j$  meet each other. The record  $ER_i$  generated by node  $i$  is formatted as follows:

$$\begin{aligned}
 ER_i &= \langle ID_i, ID_j, sn_i, sn_j, t, SL_i, RL_i \rangle \\
 MR_m &= \langle ID_m, SRC_m, DST_m \rangle \\
 SL_i &= \{MR_m \mid i \text{ sends message } m \text{ to } j\} \\
 RL_i &= \{MR_m \mid i \text{ gets message } m \text{ from } j\} \\
 ER_i^* &= ER_i, sig_i, sig_j
 \end{aligned}$$

where  $sn_i, sn_j$  are the sequence numbers assigned to the ER by node  $i$  and  $j$  respectively;  $t$  is the encounter time (in units of time slots);  $sig_i, sig_j$  are their signatures over the content of  $ER_i$  using their respective private keys.  $SL_i$  and  $RL_i$  denote the lists of message records that node  $i$  sends to and receives from node  $j$  respectively. Each message record (MR) contains three fields: the identifier of the message, the identifier of the source node who generates the message and the identifier of the destination node to whom the message is destined. A node assigns its new ER with the sequence number of its latest ER incremented by 1.

Due to possible storage limitation, each node only keeps a window of  $w$  most recent encounter records and presents them to the neighbors. Each node maintains the meeting list (ML) to store the summary information for other nodes based on their encounter histories. For each node  $j$  that  $i$  previously encountered,  $i$  creates a meeting summary (MS)  $MS_j^i$  for  $j$  which includes the identification of  $j$ , the sequence number and timestamp of the latest encounter record of node  $j$  and  $j$ 's trust reputation. Each entry  $MS_j^i$  is formatted as follows:

$$MS_j^i = \langle ID_j, latest\_sn_j, latest\_t_j, TR_j^i \rangle$$

where  $ID_j$  denotes the identifier of node  $j$ ,  $latest\_sn_j$  and  $latest\_t_j$  are the sequence number and contact time of the latest encounter record of node  $j$  known to  $i$ . Node  $i$  can learn this information when it meets either node  $j$  directly or some other node  $k$  who has a recent encounter record with  $j$ .  $TR_j^i$  represents the most updated trust reputation of node  $j$  judged locally by node  $i$ . When two nodes first meet, they can set the initial reputation  $TR_{init}$  for each other.

## 2.2 Individual Packet-Dropping Attack Model

Individual adversaries launch the attack by dropping messages and manipulating their own ER histories. An attacker first receives messages from other nodes but later drops them with a certain probability. Blackhole attacker drops all received messages (dropping probability = 100%) while greyhole attacker drops partially (dropping probability lower than 100 percent). The dropping occurs even if the attacker still has enough buffer to store messages.

To conceal its dropping misbehavior, an attacker may manipulate its own ERs history to obtain an encounter record window beneficial for itself to present to other nodes. In Fig. 2, for example, the malicious node  $m$  has an actual series of encounter records  $A, B, C, D$  with respective

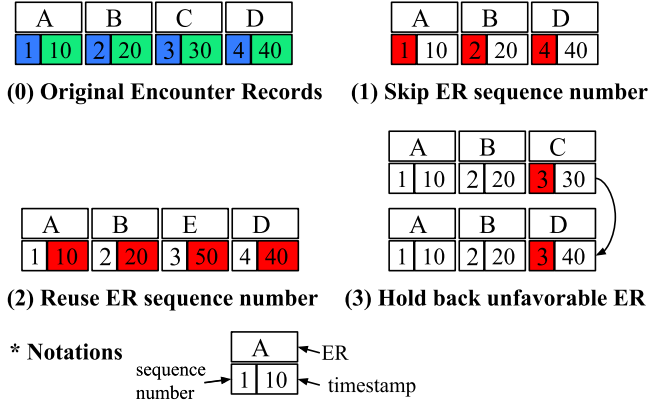


Fig. 2. Manipulation strategies of attackers.

$\langle sequence\ number, timestamp \rangle$  of  $\langle 1, 10min \rangle$ ,  $\langle 2, 20min \rangle$ ,  $\langle 3, 30min \rangle$  and  $\langle 4, 40min \rangle$ . If the adversary wants to delete the unfavorable ER  $C$  with sequence number 3, it can use one of the following three manipulation strategies:

- 1) *Skipping ER sequence number.*  $m$  deletes the ER  $C$  and does not generate any other ER using the sequence number 3.
- 2) *Reusing ER sequence number.*  $m$  deletes the ER  $C$  and uses the sequence number 3 to generate another ER  $E \langle 3, 50min \rangle$ .
- 3) *Holding back unfavorable ERs.*  $m$  uses the same sequence number 3 to generate different ERs:  $C \langle 3, 30min \rangle$  and  $D \langle 3, 40min \rangle$ . After achieving a favorable ER, it will update the sequence number to generate new ERs.

## 3 DETECTING INDIVIDUAL ATTACKERS

Individual adversaries may have two misbehaviors as presented in Section 2.2. Both misbehaviors leave some traces that we can exploit to design the defense method. When manipulating ERs, attackers violate the consistency rules of creating encounter records. When dropping messages, malicious nodes only relay parts of the messages they receive and prefer sending messages for themselves to sending for other nodes. We can design proper forwarding ratio metrics that reflect this observation to detect dropping.

The evaluation process is as follows. When two nodes  $i$  and  $j$  encounter each other, node  $j$  submits a window of latest  $w$  ERs for node  $i$  to assess its behavior and vice versa. Node  $i$  will check if node  $j$  manipulated the encounter records history or dropped messages. If node  $j$  is detected as committing either misbehaviors, node  $i$  will punish it accordingly. Each node maintains a local black list which lists malicious nodes that it detects as blackhole or greyhole attackers.

### 3.1 Manipulation of Encounter Records

According to the rule of creating ER, a series of consecutive well-behaved ERs has sequential sequence numbers. Besides, ER with higher sequence number has a bigger timestamp. However, adversaries using the first two strategies violate either rules as seen in Fig. 2. In the first strategy, the new ER history has the series of sequence number = 1, 2, 4 which is not sequential. In the second strategy, the new ER

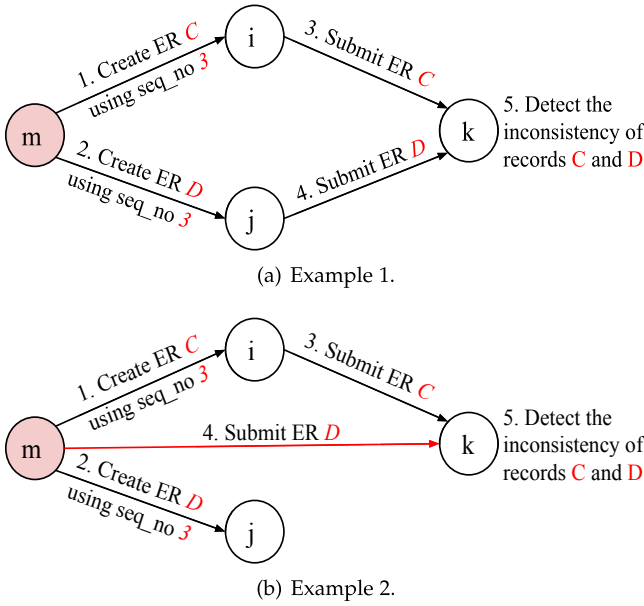


Fig. 3. Detecting inconsistent assignment of ER sequence number.

history has the series of time stamp = 10, 20, 50, 40 which is not monotonically increasing. When the manipulated ER window is presented to a neighbor node in later encounter, the misbehavior is easily detected.

The process of detecting the strategies 1 and 2 is as follows. Upon encountering node  $j$ , node  $i$  obtains  $j$ 's encounter history window and examines if the records follow the sequence and timestamp order. Besides, node  $i$  can also cross-check each node  $k$  that  $j$  encountered within the window. The encounter record between  $j$  and  $k$  will have its sequence number  $sn$  assigned by  $k$  and timestamp  $t$  be checked against  $latest\_sn_k$  and  $latest\_t_k$  recorded in node  $i$ 's meeting list. If any node violates the rules, node  $i$  will blacklist them.

In the third strategy, the produced manipulated ERs history is  $A, B, D$  which abides by both sequence number and timestamp rules. Thus the neighbor cannot discover the manipulation at all if it depends on the ER window presented by  $m$  only. However, the misbehavior leads to the inconsistency that at least two different records  $C$  and  $D$  have been assigned with the same sequence number by  $m$ . Fig. 3 illustrates how a normal node  $k$  can discover a malicious node  $m$  who uses the same sequence number 3 to create ERs  $C$  and  $D$  with two nodes  $i$  and  $j$ . In Fig. 3a, upon encountering  $i$ ,  $k$  notes and records that  $m$  has assigned the sequence number 3 to an ER with  $i$  when it processes  $i$ 's ER history. If later on,  $k$  also encounters  $j$ , it will discover that  $m$  has also assigned the sequence number 3 to the encounter with  $j$ .  $m$  is thus detected as malicious. Another variation of the detection process is shown in Fig. 3b where node  $m$  itself encounters node  $k$ . Upon meeting  $i$  and  $m$  and checking their submitted encounter history, node  $k$  can also detect that  $m$  has used the same sequence number 3 in both encounters with  $i$  and  $j$ .

We exploit this observation to detect the third strategy as follows. When each node  $i$  receives and processes an ER history of a neighbor  $j$ , it records the sequence number assignment by other nodes, not only by  $j$  but also by those encountering  $j$ . In this way, each node  $i$  has a global view of sequence number assignment over the network so that it

can inspect if any sequence number inconsistency exists. Once detected, the malicious node committing the inconsistency is blacklisted. The process does not incur additional communication overhead and only requires a small storage of sequence number history.

### 3.2 Dropping Misbehavior

Attackers might receive a lot of messages but only relay a small portion of them as the rest have been dropped intentionally. Among the messages sent out by selfish attackers, a large portion are messages generated by themselves as they drop others' messages but keep their own. Based on these observations, we define the metrics named relaying ratio  $RR$  and self-forwarding ratio  $SFR$  that can potentially reveal the dropping misbehavior as follows:

$$RR = \frac{N_{RS}}{N_{RNS}},$$

where  $N_{RS}$  is the total number of messages that  $j$  received and already relayed by forwarding to another node;  $N_{RNS}$  is the total number of messages that  $j$  received as a relay (not the ultimate destination) but  $j$  has not sent out yet,

$$SFR = \frac{N_{self\_send}}{N_{send}},$$

where  $N_{self\_send}$  is the total number of messages that are generated and sent out by node  $j$ ;  $N_{send}$  is the total number of messages sent out (regardless of the message source). It is noted that  $N_{RS}$ ,  $N_{RNS}$ ,  $N_{self\_send}$ ,  $N_{send}$  are calculated over all the encounter records in node  $j$ 's ER window.

Due to the dropping misbehaviors, malicious nodes often have lower  $RR$  and higher  $SFR$  than well-behaving nodes. After obtaining the two metrics, node  $i$  compares them with predefined thresholds to take corresponding action towards updating the reputation for node  $j$ . If  $RR$  falls below the threshold  $Th_{RR}$ , the reputation of node  $j$  judged by node  $i$  will be decreased by  $\gamma$ . If  $SFR$  exceeds the threshold  $Th_{SFR}$ , the reputation of node  $j$  is further reduced by  $\rho$ . However, if both thresholds are not violated, node  $j$  will have its reputation increased by  $\lambda$ . All the thresholds are chosen empirically using simulations.

The node whose reputation exceeds threshold  $Th_{good}$  are trusted and given higher priority to receive messages while those whose reputations fall below threshold  $Th_{mal}$  are blacklisted and isolated. Upon encountering a blacklisted node  $b$ , a normal node  $n$  will not exchange messages with it. However, node  $n$  still acquires the encounter record history of node  $b$  to obtain its measures  $SFR$  and  $RR$ . After collecting these metrics for some time and observing that the blacklisted node keeps behaving well, node  $n$  can allow node  $b$  back to its friend list but resets the initial trust reputation for  $b$  to be lower than  $TR_{init}$ . The purpose of this forgiving policy is to give the malicious nodes a second chance to change their behavior. Besides, it prevents the network from totally isolating the normal nodes who have been judged wrongly as malicious nodes.

An attacker that purely drops messages can be detected as presented above. An attacker that not only drops messages but also individually manipulates ERs might achieve a favorable ER history window to avoid



TABLE 1  
Notations

Given a node  $j$  and its encounter window  $ERW$  which includes several recent encounter records, the following parameters are defined for node  $j$  and the window  $ERW$

$N_{RS}$	Total number of messages that node $j$ has received and already relayed within its encounter window
$N_{RNS}$	Total number of messages that node $j$ has received as a relay but has not sent out yet within its encounter window
$N_{self\_send}$	Total number of messages that are generated and sent out by node $j$ within an encounter window
$N_{send}$	Total number of messages that are sent out by node $j$ within an encounter window
$RR$	Relaying ratio, defined by the ratio of $N_{RS}$ over $N_{RNS}$
$SFR$	Self-forwarding ratio, defined by the ratio of $N_{self\_send}$ over $N_{send}$
$Th_{RR}$	Threshold of relaying ratio. If node $j$ 's relaying ratio is below this threshold, its reputation is punished
$Th_{SFR}$	Threshold of self-forwarding ratio. If node $j$ 's self-forwarding ratio is above this threshold, its reputation is punished

violating the thresholds of  $RR$  and  $SFR$  metrics. However, the manipulation misbehavior can be detected by the scheme in Section 3.1.

## 4 DETECTING COLLUSION ATTACKERS

An advanced technique of blackhole and greyhole attackers is that they collude with each other to conceal their misbehaviors. In the presence of collusion, the detection of individual attack suggested in Section 3 is not effective any more. In this section, we first present the collusion attack model to illustrate how colluders cooperate to avoid the detection. We then explain the proposed scheme to deal with the collusion. Table 1 summarizes the notation of the parameters.

### 4.1 Collusion Packet-Dropping Attack Model

In collusion attack, adversaries cooperate so that they can counter against our individual detection scheme proposed in Section 3 and avoid being detected as individual attackers. We assume that colluding attackers consistently misbehave over time without switching their behavior between malicious and normal states. We will discuss behavior-switching attack in Section 4.4.

The strategy for an adversary is creating fake encounter records with its colluders to manipulate its own metrics  $RR$  and  $SFR$ . Colluding attackers are assumed to know the private keys of one another and freely use them to sign fake encounters to make them appear authentic. When the malicious nodes need to forge encounters, they need to determine the parameters for the fake encounters such as number of sent and received messages, encountered peer ID.

Suppose a malicious node  $m$  currently has its ER window denoted as  $ERW = \{ER_1, \dots, ER_w\}$ . Node  $m$ 's corresponding parameters  $N_{RNS}$ ,  $N_{RS}$ ,  $N_{send}$ ,  $N_{self\_send}$ ,  $RR$ ,  $SFR$

can be inferred from  $ERW$  as shown in Section 3. A malicious message-dropping attacker is likely to have its metrics  $RR$  and/or  $SFR$  violate the thresholds:

$$RR = \frac{N_{RS}}{N_{RNS}} < Th_{RR}$$

$$SFR = \frac{N_{self\_send}}{N_{send}} > Th_{SFR}.$$

The violation of either rule causes  $m$ 's reputation to be punished by others. To avoid punishment,  $m$  needs to forge an ER so that the dropped portion is compensated and the new metrics over all its manipulated ER history are out of the abnormal range defined by the thresholds.

Denote the newly forged encounter record's parameters as: number of messages sent for itself  $n_{self\_send}$ , number of sent messages  $n_{send}$ , number of received messages  $n_{recv}$ . Suppose this fake ER makes the total number of messages received but not relayed (over the new encounter record window) increase by  $n_{RNS}$  and the number of messages received and relayed increase by  $n_{RS}$ . To hide the violating metrics,  $m$  has to choose the parameters such that:

$$RR' = \frac{N_{RS} + n_{RS}}{N_{RNS} + n_{RNS}} \geq Th_{RR} \quad (1)$$

$$SFR' = \frac{N_{self\_send} + n_{self\_send}}{N_{send} + n_{send}} \leq Th_{SFR}. \quad (2)$$

Attacker first needs to choose  $n_{RS}$ ,  $n_{RNS}$ ,  $n_{self\_send}$  and  $n_{send}$  for the forged record to satisfy the conditions of Eq. (1) and (2); then it creates fake sent and received lists,  $SL$  and  $RL$ , to fit those chosen parameters.

Next, the attacker needs to choose the encounter peer node for the forged record if it has multiple colluders. It can choose the fake encountered peer as the one that it creates the least number of forged encounters in its ER history window. In this way, the misbehaving node can minimize the encounter frequency with each colluder and avoid raising suspicion from the network. Finally,  $m$  can insert the signatures of itself and the chosen colluder over the fake encounter record.

We will show that, to satisfy the conditions of Eq. (1) and (2), the number of sent messages in forged record  $n_{send}$  must be lower bounded. First we assume node  $m$  simply sets  $n_{self\_send}$  to 0 and  $n_{recv}$  to 0. As a result,  $n_{RNS} = 0$  and  $n_{RS} \leq n_{send}$ . The fake metrics  $RR'$  and  $SFR'$  are formulated as:

$$RR' = \frac{N_{RS} + n_{RS}}{N_{RNS}} \geq Th_{RR},$$

$$SFR' = \frac{N_{self\_send}}{N_{send} + n_{send}} \leq Th_{SFR}.$$

This is equivalent to:

$$n_{RS} \geq Th_{RR} \times N_{RNS} - N_{RS}$$

$$n_{send} \geq N_{self\_send} / Th_{SFR} - N_{send}.$$

As  $n_{send} \geq n_{RS}$ , we get the lower bound for  $n_{send}$

$$n_{send} \geq \max(Th_{RR} \times N_{RNS} - N_{RS}, N_{self\_send} / Th_{SFR} - N_{send}). \quad (3)$$

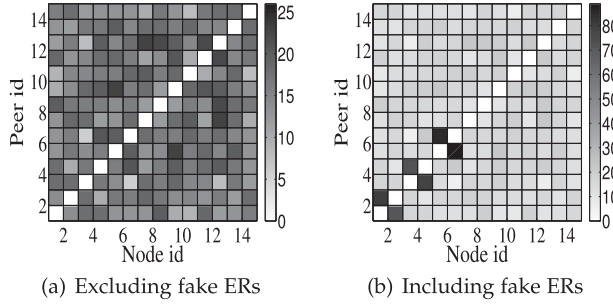


Fig. 4. Encounter records frequency between node pairs.

If  $m$  sets  $n_{self\_send}$  and  $n_{recv}$  larger than 0,  $n_{send}$  even has to be larger.

## 4.2 Analysis of Collusion Behavior

The collusion attack presented above will nullify the individual detection scheme proposed in Section 3. We need to explore the measures that possibly reveal the collusion behavior. If the attacker keeps dropping received messages, its metrics  $RR$  and  $SFR$  will be degraded continuously. To avoid being detected, it needs to keep promoting the metrics by creating fake encounters persistently over time. Besides, the adversary has a low total number of relayed messages in authentic encounters. To raise this amount and achieve well-behaving metrics, it has to set the forged number of sent messages higher than a lower bound as shown in Eq. (3). Therefore, we identify the appearance frequency and the number of sent messages of forged encounter records as the potential metrics to distinguish them from authentic records. We will investigate those two metrics and show that colluding attackers cannot hide the abnormality of both metrics at the same time.

To study the metrics, we run a simulation which involves 15 mobile nodes whose identifiers are numbered from 1-15. The first six nodes are chosen as attackers and they pair to collude (following the collusion model described earlier): (node 1-2), (node 3-4), (node 5-6). In Fig. 4, we plot the global view of encounter frequencies of all node pairs in the color matrix. The darker square means a higher value of encounter frequency for the corresponding pair. Fig. 4a shows the case where fake ERs forged by colluding pairs are excluded from computing the ER frequency. In general, it is hard to distinguish malicious pairs from normal ones. In Fig. 4b, however, malicious pairs can be easily identified when fake ERs are included in computing the ER frequency. A similar trend can be found in Fig. 5 which plots the global view of average number of sent messages. The darker

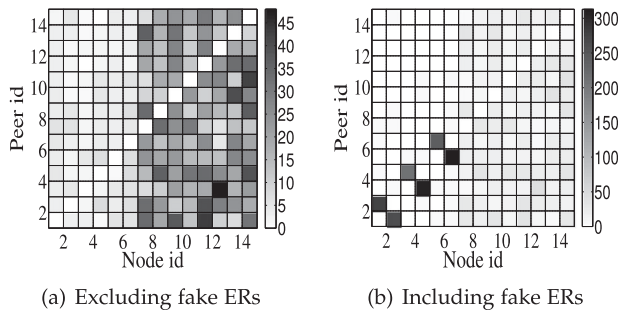


Fig. 5. Average number of sent messages between node pairs.

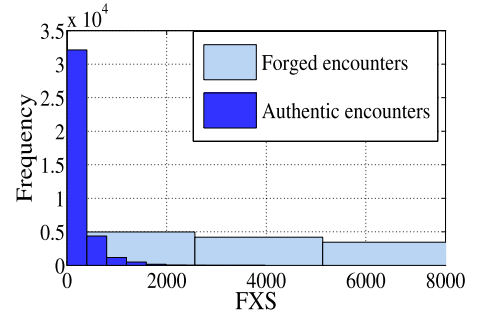


Fig. 6. FXS metric of forged ER versus authentic ER.

square means a higher value of sent messages number for the corresponding pair. Hence, encounter frequency and number of sent messages can be used as promising metrics for collusion detection.

It is noted that the collusion attackers cannot manipulate fake ERs and hide the abnormal pattern of both metrics at the same time. A malicious node can manipulate its forwarding ratio by creating one single fake ER with number of sent messages  $n_{send}$  and the risk is that  $n_{send}$  might be suspiciously high. Alternatively, the attacker can create more than one fake records, say  $l$  fake ERs with the corresponding number of sent messages  $n_1, \dots, n_l$ . As both ways are equivalent to promoting the metrics, we have:

$$n_{send} = n_1 + \dots + n_l = \sum_{i=1}^l n_i.$$

Using the latter option, the attacker can successfully reduce the number of sent messages per fake encounter (i.e.  $n_1, \dots, n_l < n_{send}$ ) to a non-suspicious range. However, it has to raise the number of fake encounters from 1 to  $l$  instead.

Based on this observation, we suggest a combined metric that accounts for both the number of sent messages and the encounter frequency as follows. Given an encounter window  $ERs$  of node  $j$ , for each node  $k$  recorded as encountering  $j$  in  $ERs$ , denote  $freq_j^k$  as the number of encounters between  $j$  and  $k$  appearing in  $ERs$ ,  $send_j^k$  as the total number of messages sent from  $j$  to  $k$  over the window. The metric  $FXS$  is defined as

$$FXS_j^k = freq_j^k \times send_j^k.$$

If  $j$  and  $k$  are colluders,  $FXS_j^k$  reflects the abnormality of fake encounters between them; thus  $FXS_j^k$  should be abnormally high. This is demonstrated in Fig. 6 which shows the histogram distribution of  $FXS$  in forged encounters and authentic encounters. As seen in the figure, most authentic encounters have  $FXS$  as low as 1,000 while forged encounters normally have  $FXS$  as high as 2,000 or more. The metric  $FXS$  of forged encounters is often much higher and can be differentiated from that of authentic encounters using threshold.

## 4.3 Detection of Collusion

We make use of the above analysis to design the collusion detection scheme. When node  $i$  encounters node  $j$ , node  $i$  checks if node  $j$  has launched the collusion message-dropping attack in a 2-phase process: suspicion and

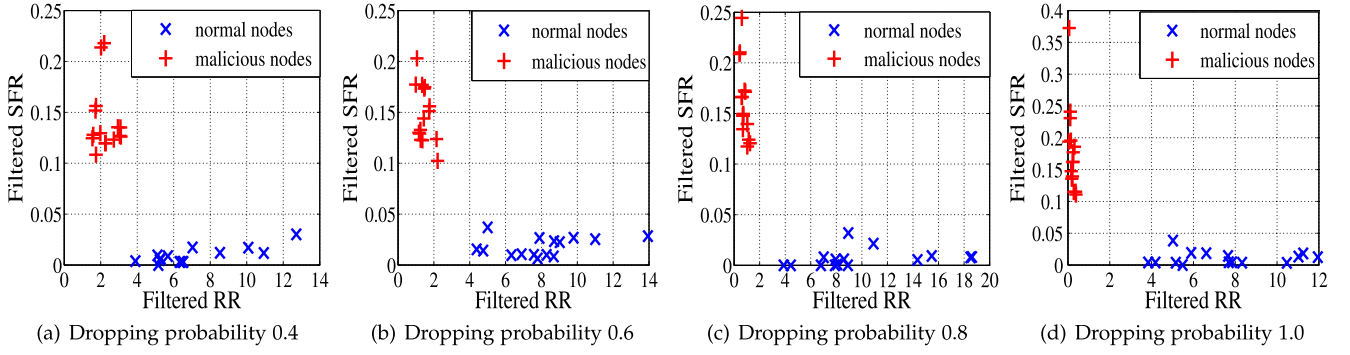


Fig. 7. Filtered metrics of nodes in suspicious list.

confirmation. The first phase produces a list of suspicious colluders based on the  $FXS$  metric. The second phase is required to avoid false accusation of the suspected node who has high  $FXS$  metric because it actually has many encounters and exchanges many messages with node  $j$ .

**Suspicion.** In the first step, node  $i$  uses the ER history window provided by node  $j$  to infer the list of nodes that  $j$  has met in the window and the corresponding  $FXS$  metric. If the metric  $FXS_j^k$  exceeds a defined threshold  $Th_{FXS}$ , node  $k$  will be added to the suspicious list as it possibly colludes with  $j$ . Once suspected, step 2 is triggered to start the verification process.

**Confirmation.** In this step, the ERs of node  $j$  along with each node  $k$  in the suspicious list will be filtered from the current ERs window. Then the evaluation process described in Section 2.2 is implemented on the newly obtained ERs window, i.e., the suspiciously fake records will not be considered in the computation. This filtered window will have its relaying ratio  $RR'$  and own forwarding ratio  $SFR'$  calculated and checked against the threshold  $Th_{RR}$  and  $Th_{SFR}$ . If the thresholds are violated, the same penalty is meted out to node  $j$  and to the corresponding node in the suspicious list.

Fig. 7 shows how the filtered  $RR$  and  $SFR$  are distributed for malicious nodes and normal nodes in the suspicious list. As seen in the figure, malicious nodes have lower  $RR$  and higher  $SFR$  than normal nodes though all of them get suspected due to abnormal  $FXS$  metric. It demonstrates that the filtered metrics can be well exploited to differentiate malicious and normal nodes.

We summarize how the collusion detection is integrated with the individual detection in Algorithm 1.

#### 4.4 Special Case: Collusion Behavior-Switching Attack

To reduce the false positive rate, we incorporate the reputation approach into our detection scheme. Given a node  $n$  encountering a node  $k$ , if  $n$  judges  $k$  as well-behaving,  $n$  rewards  $k$  by increasing its reputation. Otherwise,  $n$  punishes  $k$  by reducing its reputation instead of immediately blacklisting  $k$ . Only if  $n$  could observe the repeated misbehavior of  $k$  over time which leads to very low reputation of  $k$ , then  $n$  will blacklist  $k$ .

However, malicious nodes could also make use of this feature to bypass our detection as follows. Suppose a normal node takes  $T$  time units to observe and blacklist an adversary. Colluding adversaries first attack for  $T$  time

units before being blacklisted, then behave well for  $T_2$  time units to regain their reputation before continuing to attack. We call this misbehavior the collusion behavior-switching attack.

#### Algorithm 1. Algorithm of node $i$ evaluating node $j$

**Require:** current ER window of node  $j = ERW$

**Require:**  $dropping = \text{false}$ ,  $collusion = \text{false}$

**Require:** parameters  $Th_{RR}$ ,  $Th_{SFR}$ ,  $Th_{FXS}$ ,  $\gamma$ ,  $\rho$ ,  $\lambda$  {individual attacker detection}

```

1: Calculate the metrics  $RR$  and  $SFR$  from  $ERW$ 
2: if ( $RR < Th_{RR}$ ) then
3:    $TR_j^i = TR_j^i - \gamma$ 
4:    $dropping = \text{true}$ 
5: end if
6: if ( $SFR > Th_{SFR}$ ) then
7:    $TR_j^i = TR_j^i - \rho$ 
8:    $dropping = \text{true}$ 
9: end if {cooperating attacker detection}
10:  $p = c_1$ 
11: for each node  $k$  that  $j$  encounter in  $ERW$  do
12:    $freq_k$  = number of times  $k$  and  $j$  encounters over  $ERW$ 
13:    $send_k$  = number of messages  $j$  sends to  $k$  over  $ERW$ 
14:    $FXS_k = freq_k \times send_k$ 
15:   if ( $FXS_k > Th_{FXS}$ ) then
16:      $ERW' = ERW$  excluding encounter records with  $k$ 
17:     Calculate the metrics  $RR'$  and  $SFR'$  from  $ERW'$ 
18:     if ( $RR' < Th_{RR}$ ) then
19:        $TR_j^i = TR_j^i - \gamma$ 
20:        $TR_k^i = TR_k^i - \gamma$ 
21:        $collusion = \text{true}$ 
22:     end if
23:     if ( $SFR' > Th_{SFR}$ ) then
24:        $TR_j^i = TR_j^i - \rho$ 
25:        $TR_k^i = TR_k^i - \rho$ 
26:        $collusion = \text{true}$ 
27:     end if
28:   end if
29: end for
30: if ( $dropping == \text{false}$ ) and ( $collusion == \text{false}$ ) then
31:    $TR_j^i = TR_j^i + \lambda$ 
32: end if

```

It is noted that this behavior-switching attack is theoretically feasible but hard to be implemented. The detection time  $T$  is different for different nodes as it depends on their respective encounters with the malicious nodes. No doubt,

TABLE 2  
Detection Rate of ER Manipulation

EMR	0.05	0.1	0.2	0.3	0.4
Detection rate	0.98	0.99	1.0	1.0	1.0

the malicious nodes can estimate  $T$  but they cannot ensure the accuracy of  $T$ . Hence the malicious nodes are still liable to be detected by some normal nodes.

To further suppress this attack, we design the scheme such that reputation is hard to build but easy to lose. We set reward smaller than punishment so that adversaries will have their reputations dropped more quickly than increased. Meanwhile, behavior-switching attackers need to gain back the amount of reputation as much as the amount they lose. As a result, they have to well-behave (for  $T_2$  time units) longer than the attack period (for at most  $T$  time units), i.e.  $T_2 > T$ . Thus the attack frequency is mitigated. In Section 5.5, we present simulation results to show the mitigation effect of our scheme against this attack.

## 5 PERFORMANCE EVALUATION

### 5.1 Simulation Setup

The simulation is implemented in ONE simulator [16]. The simulated network contains 40 nodes which can communicate in ad-hoc mode using Wi-fi in a transmission range of 100 meters. They travel over an area of 4,500 m  $\times$  3,400 m, at speeds of 10-50 km/h, using Shortest Path Map Based Movement Model which is available in ONE to simulate the movement of vehicles on the streets. The simulation time is 43,200 seconds (12 hours). Messages are generated at the rate of one per 25-30 seconds. The message size is in the range of 50 kB-1 MB. This setting is applied to all the following experiments. For each experiment, the simulation runs for 10 times with random seeds and the average of the measured metrics are recorded and presented.

### 5.2 Evaluating the Detection Performance of SDBG

We assess the detection performance of SDBG in two aspects: detecting ER manipulation and detecting collusion dropping. We use the following metrics for evaluation:

- Detection accuracy: percentage of malicious nodes that can be detected by normal nodes
- Detection delay: the time taken for the misbehavior to be detected.
- Detection false positive rate: percentage of normal nodes that are mistakenly judged as malicious by other normal nodes.

#### 5.2.1 Detection of ER Manipulation Misbehavior

The total number of attackers is set to 12. Attackers hold back unfavorable ERs to hide their dropping behaviors, as described in Section 3.1. We only consider this manipulation strategy because it is the most challenging to detect among three strategies. We define ER manipulation ratio (EMR) as the ratio of the number of manipulated ERs over all ERs for a malicious node. We vary attackers' EMR from 0.05 to 0.4.

Table 2 shows the percentage of normal nodes that can detect malicious nodes. We find that most of the malicious

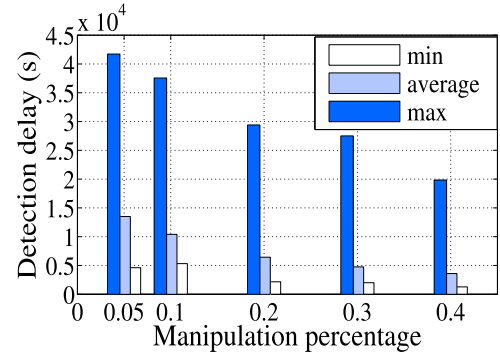


Fig. 8. Detection delay of ER manipulation.

nodes can be accurately detected by all normal nodes. Fig. 8 illustrates three measures of detection delay: (i) min delay is the time taken for all malicious nodes to be detected at least once by any normal node; (ii) average delay is the mean delay over all normal nodes that can detect the misbehavior; (iii) max delay is the time that all malicious nodes have been detected by all the normal nodes. The detection delay is always highest at the manipulation percentage of 0.05 and lowest at the manipulation percentage of 0.4. When the manipulation percentage increases, the detection delay is reduced and detection rate is increased to 100 percent. This means the more frequently the attackers manipulate the encounter records, the faster and more possibly they can be discovered.

#### 5.2.2 Detection of Collusion Packet Dropping Misbehavior

In this section, we evaluate the detection of collusion dropping under (1) varying attacker setting (number of attackers, collusion setting, dropping probability) and (2) varying SDBG setting ( $Th_{RR}$ ,  $Th_{SFR}$  and  $Th_{FXS}$ ).

*Varying attacker setting.* The total number of malicious nodes is set to 6 and 12, in which they form collusions of two, four and six members. For example, if 12 attackers form collusions of four members, this case is represented as 4x3, i.e., there are three groups of attackers and each group consists of four colluding nodes. The dropping probability is varied from 0.4 to 1. The routing protocols used are Prophet [17], MaxProp [18] and Spray and Wait [19]. The thresholds  $Th_{RR}$ ,  $Th_{SFR}$  and  $Th_{FXS}$  are fixed for each routing protocol.

Fig. 9 presents the detection accuracy of SDBG with varying routing protocols and attackers settings. From the figure, we have the following findings: (a) for all routing protocols, SDBG achieves more than 70 percent accuracy, indicating good overall accuracy performance of SDBG; (b) when the number of colluders per group increases, the detection rate drops. Having more colluders helps a malicious node to distribute its fake ERs and hence lower the fake encounter frequency with each colluder; (c) when the dropping probability decreases, the detection accuracy also drops. Less suspicion is aroused if a malicious node reduces its dropping behavior.

We plot the false positive rate of SDBG under varied contexts in Fig. 10. From this figure, we find that for all routing protocol, SDBG achieves a false positive rate lower than 1.6



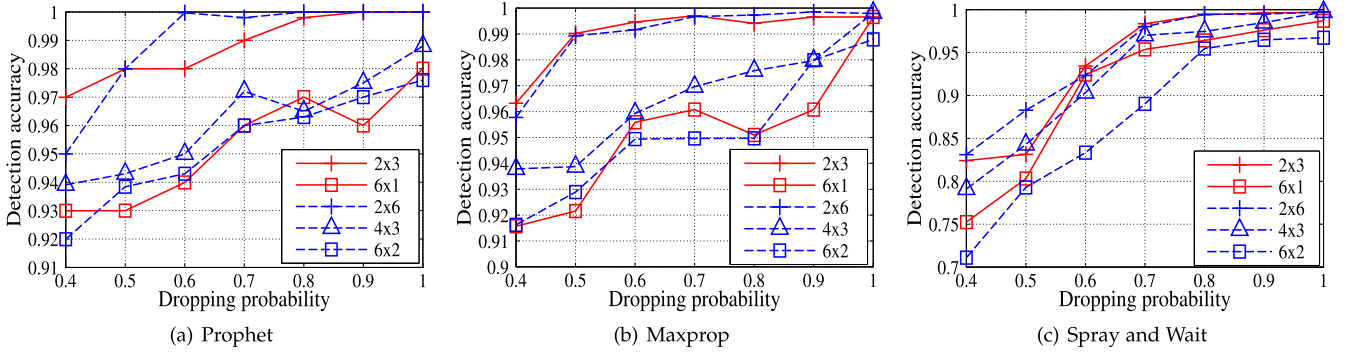


Fig. 9. SDBG's detection accuracy under varying attack settings.

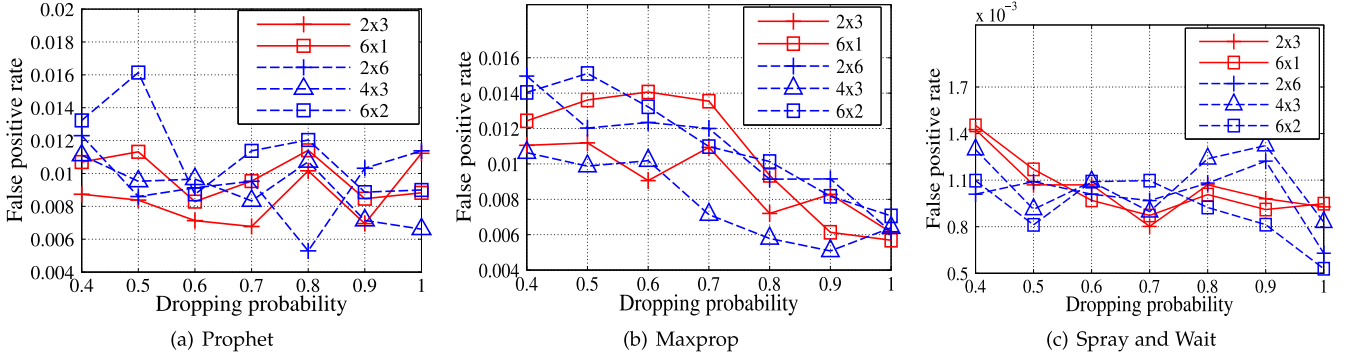


Fig. 10. SDBG's false positive rate under varying attack settings.

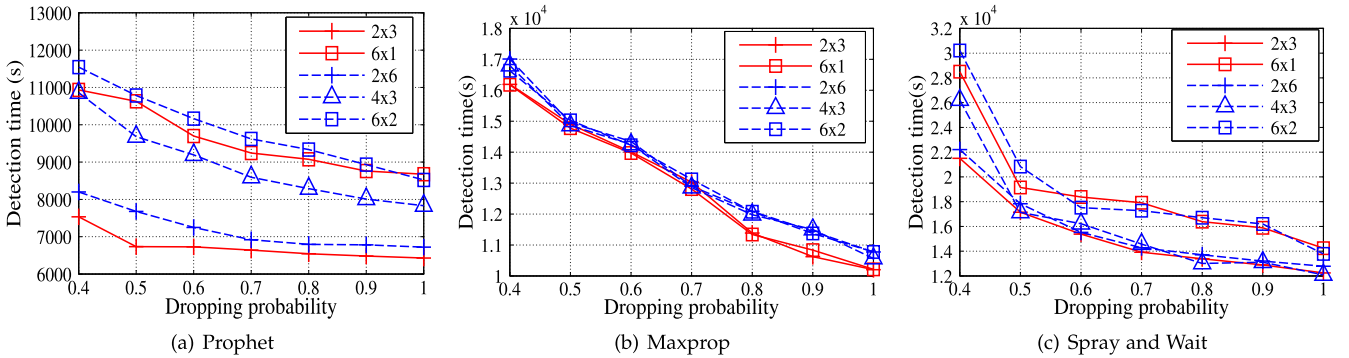


Fig. 11. SDBG's detection time under varying attack settings.

percent. This shows that our two-phase scheme works efficiently in reducing the false positive.

The minimum detection delay under the various collusion settings are plotted in Fig. 11. We find that (a) for all routing protocols, the delay decreases when the dropping probability increases. Specifically, greyhole attackers takes longer time to be detected than blackhole attackers; (b) when the number of colluders per group increases, the delay also increases since it becomes more challenging to figure out a bigger group of colluders.

*Varying thresholds setting.* We fix the attacker settings as follows: 12 attackers are divided into two groups of six colluders, the packet dropping probability is set to 0.5, the routing protocol is chosen to be Prophet. We vary only one of the three system parameters ( $Th_{SFR}$ ,  $Th_{RR}$ ,  $Th_{FXS}$ ) in SDBG at a time while the other two are fixed.

The results are shown in Figs. 12 and 13. As seen in Figs. 12a and 13a, when  $Th_{RR}$  is increased, the detection

accuracy is increased from 0.92 to 0.95 and the max detection time is reduced from 22,000 to 20,500 seconds; but correspondingly, the false positive rate is increased from 1 to 3 percent. This is because malicious nodes tend to have lower  $RR$  than normal nodes. If  $Th_{RR}$  is raised, there is more chance that malicious nodes violate the thresholds and get detected earlier but at the same time more normal nodes are likely to be misjudged.

Similarly, the trend of simultaneously decreased detection accuracy, increased detection time and decreased false positive is observed as  $Th_{SFR}$  and  $Th_{FXS}$  increase. This demonstrates the trade-off among the three different performance metrics when the thresholds are varied.

### 5.3 Comparing with Other Schemes

We compare our scheme, SDBG, with two state-of-the-art schemes, MDS [14] and Li's scheme [15] in the overall

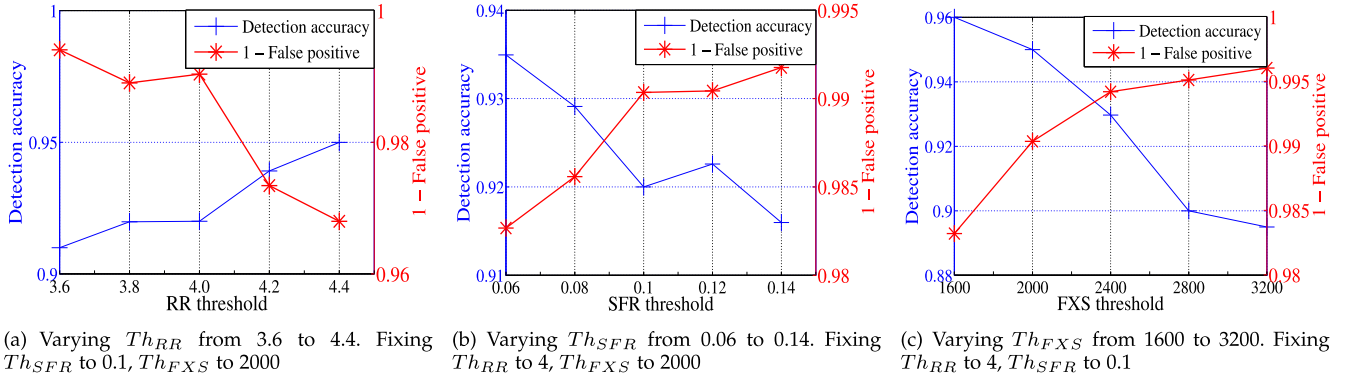


Fig. 12. Detection accuracy and false positive in Prophet under varying threshold settings.

network performance when they are used to detect and prevent message dropping attackers. We use the following metrics for comparison:

- Number of wasted transmissions: number of messages that malicious nodes have received from normal nodes and then dropped intentionally.
- Delivery ratio: percentage of messages delivered to destinations out of total generated messages
- Delivery delay: average time taken for a message to reach the destination

We briefly summarize how the other two schemes work. Similar to SDBG, MDS suggests statistical metrics based on encounter history to judge nodes but it does not consider dealing with collusion attacks. In Li's scheme, a node is judged as dropping messages if its current message buffer does not include all the messages it receives and stores in its previous contact. In collusion attack, an adversary can forge buffer by creating fake previous contact in which it claims it did not store or receive the dropped messages. Li's scheme addresses this collusion behavior by requiring each node to keep multiple recent contact records and present them for judgement. In this way, a normal node can see a longer contact history of a malicious node which likely includes an authentic encounter, and thus observe the messages it actually receives.

These schemes are evaluated under different scenarios where individual attacks or collusion attacks are launched. The number of attackers are set to 12. In individual attack, malicious nodes attack separately and cannot forge encounter records. In collusion attack, 12 adversaries are divided

into three groups, each of which contains four colluders. The dropping probability is varied from 0.4 to 1. The routing protocols are Prophet and MaxProp.

### 5.3.1 Countering Individual Attack

Fig. 14 shows the overall network performance when using different schemes to detect individual attack. As seen in Fig. 14a, MDS can achieve the number of wasted transmissions of at most 33,000 and 9,000 for Prophet and Maxprop respectively. The number of wasted transmission in SDBG is at most 31,000 for Prophet and 5,000 for Maxprop. SDBG always outperforms MDS in both routing protocols at any dropping probability. It shows that the suggested forwarding ratio metrics of SDBG can work better than those of MDS.

It can be observed that in MDS and SDBG, the plot of number of wasted transmissions has a valley point at the dropping probability of 0.8. At low dropping probability, the percentage of dropped messages is low but the detection accuracy is also low, providing adversaries more time to continue their attack. When the dropping probability is increased, the number of dropped messages is higher but the adversaries are detected faster and more accurately.

As seen in Fig. 14b, MDS and SDBG are comparable in keeping the average delivery latency of at most 670 seconds for Prophet and at most of 600 seconds for Maxprop. The delivery latency in Li's scheme is at least about 1,200 seconds, which is about two times higher than that of MDS/SDBG. Based on the results in Figs. 14a and 14b, Li's scheme can reduce the number of wasted transmission more significantly than MDS/SDBG but the tradeoff is that its delivery

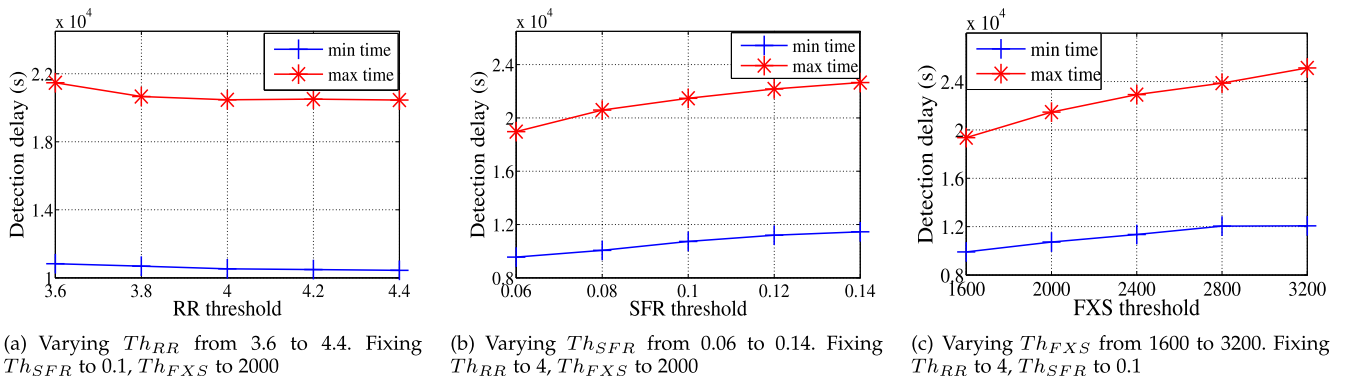


Fig. 13. Detection time of Prophet under varying threshold setting.

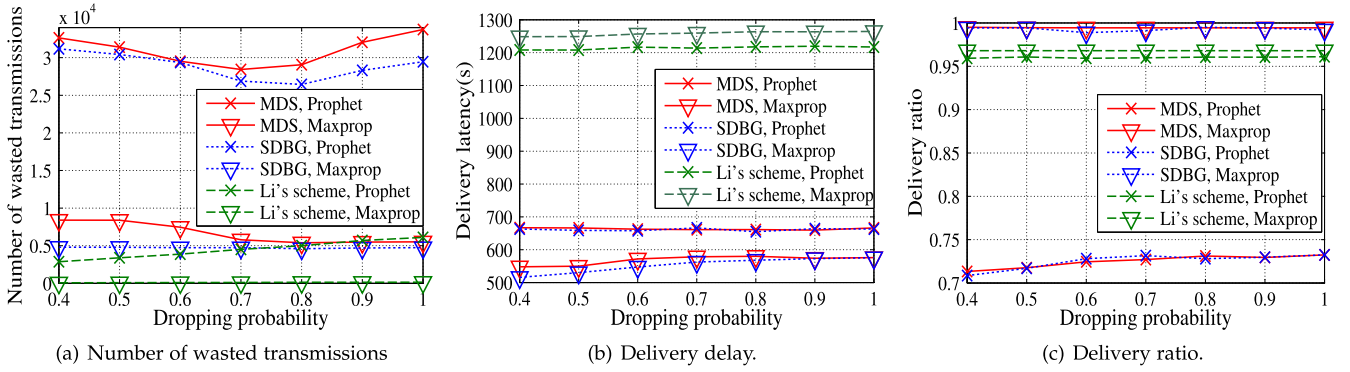


Fig. 14. SDBG versus other schemes for individual attack.

latency is much longer. Li's scheme cannot differentiate the dropping behavior of malicious nodes from normal nodes who drop with valid reasons. As a result, Li's scheme produces high false positive rate, making nodes miss many transfer opportunities with good relays and thus adds more delay to the message delivery time.

As seen in Fig. 14c, the delivery ratio of MDS and SDBG are not significantly different, which is at least 0.72 for Prophet and at least 0.99 for Maxprop. Li's scheme can sustain the delivery ratio of about 0.96 for both Maxprop and Prophet. MDS and SDBG can achieve higher delivery ratio in Maxprop but lower delivery ratio in Prophet than Li's scheme. Compared to Maxprop, Prophet floods more messages for relay per encounter, leading to inefficient use of buffer storage. In Prophet, Li's scheme reduces a lot of relays due to its high misjudge rate, making the use of buffer more efficient and achieving a higher delivery ratio.

### 5.3.2 Countering Collusion Attack

Fig. 15 plots the network performance in the context of collusion attack. As seen in Fig. 15a, the number of wasted transmissions of MDS is in the range of [90,000-180,000] in Prophet and [13,000-30,000] in Maxprop. As MDS cannot detect the collusion attack at all, attackers are not prevented and the number of wasted transmissions thus increases linearly with the dropping probability. Whereas in SDBG, the amount of wasted transmissions can be kept at 18,000 and 8,000 in Prophet and Maxprop respectively. As SDBG can detect colluders accurately, the number of wasted transmissions is kept stable with varying dropping probabilities.

SDBG can reduce the wasted transmissions by at least 1.5 x compared to MDS and works especially efficiently in the presence of high dropping-probability attackers.

As seen in Fig. 15b, the average delivery latency of both MDS/SDBG is at most 630 seconds and 500 seconds in Prophet and Maxprop. As seen in Fig. 15c, the delivery ratio of MDS is lower than 0.7 in Prophet and about 0.99 in Maxprop. Whereas, the delivery ratio of SDBG is above 0.7 in Prophet and about 0.99 in Maxprop. It is observed that SDBG incurs a larger or equal delivery ratio and shorter delay compared to MDS but the difference is not so significant. Prophet and Maxprop are both replication-based routing protocols which allow a packet to be replicated and transmitted in multiple paths. Even if an attacker drops a packet in one path, the packet still has a chance to reach the destination via other paths. As a result, though MDS cannot detect and exclude collusion dropping attackers, it still achieves a comparable delivery ratio with SDBG.

In Li's scheme, the number of wasted transmissions is the lowest and the delay latency is the worst among the three schemes, which is also attributed to the high misjudgement rate. Although Li's scheme can deal with colluding adversaries, the high false positive rate results in the under utilisation of network connectivities. In contrast with Li's scheme, SDBG can detect attackers with high accuracy and low false positive. Most normal nodes are not excluded from the network by mistake so that the contact opportunities are exploited more efficiently and the average delivery delay is lower than Li's scheme. The number of wasted transmissions in SDBG is higher than Li's scheme within the simulation time because it needs more time to accurately

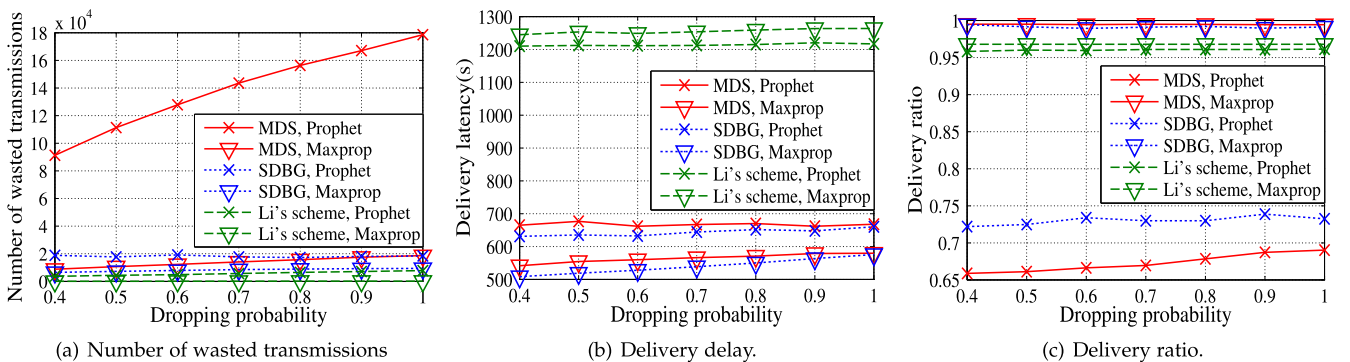


Fig. 15. SDBG versus other schemes for collusion attack.

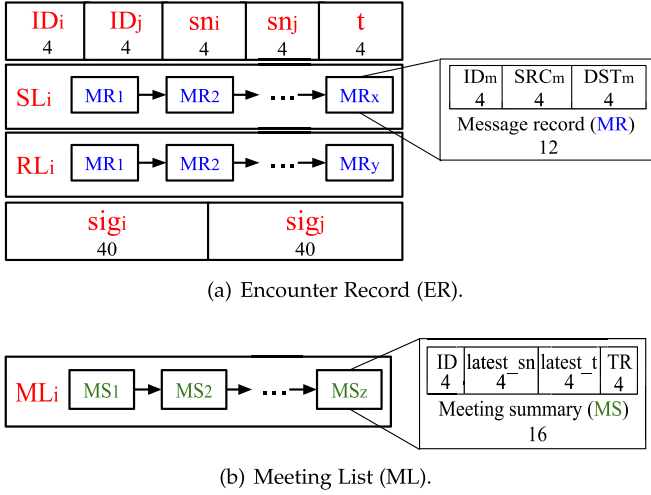


Fig. 16. Size of data overhead.

judge the misbehavior. However, in the long term, SDBG can reduce the number of wasted transmissions as efficiently as Li's scheme because the malicious nodes are ultimately blacklisted correctly.

#### 5.4 Storage Overhead

There are two types of storage overhead: temporary and permanent. The temporary overhead is the encounter histories of other nodes that are stored temporarily and deleted immediately after the evaluation. The permanent overhead for each node includes its own ER history and meeting list. We only consider the permanent storage overhead. As each node needs to keep  $w$  most recent records in its history, the ER history overhead is:

$$Overhead_{ER} = w \times size_{ER}. \quad (4)$$

Fig. 16a shows an overview of the fields in the ER and their corresponding sizes in bytes. The number of message records in sending list  $SL$ /receiving list  $RL$  is estimated as the average number of sent/received messages per ER. We run a simulation and obtain the average number of records as 20 for Prophet routing protocol. As a result, the average size of ER is  $4 \times 5 + 20 \times 12 + 20 \times 12 + 40 \times 2 = 580$  B. As  $w$  is set to 100 by default, the total storage overhead of ER history is  $100 \times 580$  B = 58 kB.

Each node also stores the meeting summary of all its neighbors in ML, thus the ML overhead is:

$$Overhead_{ML} = n \times size_{MS},$$

where  $n$  is the number of neighbors. For our simulation setting,  $n$  is 39. As seen in Fig. 16b, the MS size is 16 B. The total ML overhead is  $39 \times 16 = 624$  B, which can be ignored compared to the ERs overhead (58 kB). Therefore, we can consider the total storage overhead as being equivalent to the ER overhead which is 58 kB. This total amount of overhead is affordable for mobile devices such as mobile phones and negligible for notebooks.

##### 5.4.1 Choosing Window Size

According to Eq. (4), ER history overhead increases linearly with the window size  $w$ . Thus it is necessary to reduce  $w$  for

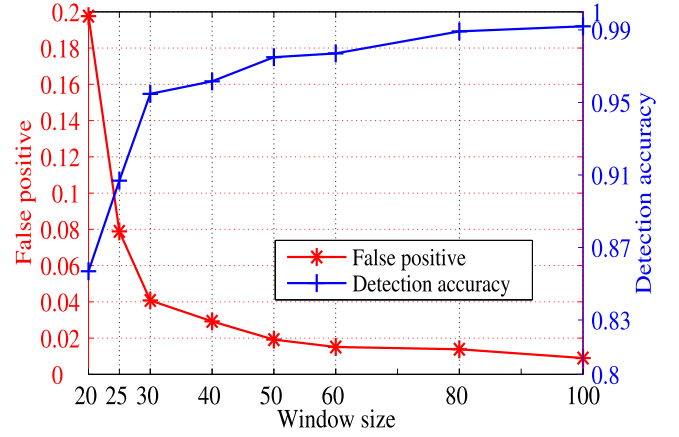


Fig. 17. SDBG detection performance under varying window size.

nodes with very limited storage to reduce their overheads. However, there is a trade off between the SDBG's detection performance and the window size. We plot the detection accuracy and false positive over varied window size in Fig. 17. The simulation setting is three groups of attackers with four colluders in each group and the dropping probability is 0.6. Prophet routing protocol is used.

Fig. 17 shows that the larger the window, the higher is the detection accuracy and the lower is the false positive rate which SDBG can achieve. When  $w$  increases, the detection accuracy experiences a sharp increase at  $w$  of 30, then stabilizes thereafter. A similar trend is observed for the false positive rate. This fact allows nodes to choose smaller  $w$  that can reduce the storage overhead significantly without sacrificing the SDBG performance a lot. For example, devices with very limited storage can choose a medium window size such as 30. This reduces the storage overhead by 1/3 compared to using window size of 100 but only degrades the detection accuracy and false positive a bit compared to the best SDBG performance. Nodes can choose an appropriate window size that fits their storage availability and their requirements on SDBG performance.

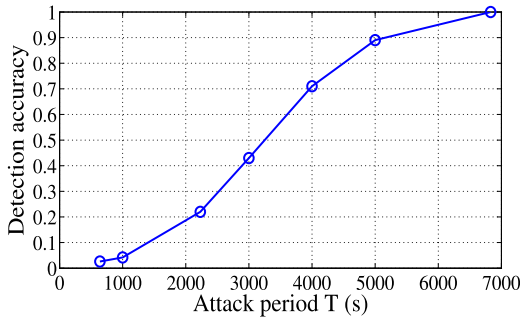
#### 5.5 Mitigating Collusion Behavior-Switching Attack

We run simulations to demonstrate the effect of mitigating the collusion behavior-switching attack (described in Section 4.4) using SDBG. The simulation settings are as follows. There are three groups of attackers with two colluders in each group. The dropping probability is 1.0. The window size is 100. The routing protocol is Prophet. The number of runs is 10.

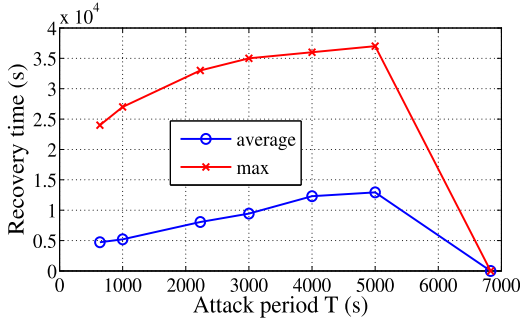
First, we set the malicious nodes to attack for a period of  $T$  and to behave well for the rest of the simulation time. We choose  $T$  based on the results of detection time in Fig. 11a for the settings above. The average detection time over 10 runs is 6,829 s. For each run, we measure the minimum detection time which is the shortest time to detect at least one malicious node. After 10 runs, we obtain the minimum detection time as 639 s and the average minimum detection time as 2,230 s. We vary  $T$  from the minimum detection time to the average detection time.

Fig. 18a shows the average detection accuracy of SDBG over attack period  $T$ . We observe that even when  $T$  is set to the minimum detection time, the detection accuracy is low





(a) Detection accuracy.



(b) Recovery time.

Fig. 18. Varying attack period  $T$ .

(2 percent) but still larger than 0. When  $T$  is set to the average detection time, the detection accuracy can reach 100 percent. This is because even when the attackers stop attacking, parts of their misbehaviors are still reflected in their ER histories. Thus, nodes that encounter the attackers early after the attack period might detect the malicious nodes.

After the attack period  $T$ , the reputations of the malicious nodes drop and they need to behave well for  $T_2$  to recover their reputations to the initial value  $TR_{init}$ . Fig. 18b plots the average and maximum recovery time over  $T$ . It is noted that when  $T$  is set to the average detection time (6,829 s), the recovery time is 0 as all attackers are detected and cannot recover their initial reputations at all. Besides, the average recovery time is 3-6x larger than the attack period  $T$ . This demonstrates that our reward/punishment scheme can effectively mitigate the collusion behavior-switching attack.

Next, we set the malicious nodes to alternate between bad and good behaviors for  $T$  and  $T_2$  respectively. For each value of  $T$ , we set  $T_2$  to the average and maximum recovery time obtained in Fig. 18b. Fig. 19 shows the detection accuracy over  $T$  under such setting. When  $T_2$  is set to the average recovery time, the detection accuracy is larger than 60 percent. Even when  $T_2$  is set to the maximum recovery time, it is still possible for the attackers to be detected. Hence, to avoid being detected, attackers have to decrease  $T$  and increase  $T_2$ , which in turn leads to insufficient attacking periods. Moreover, the result also demonstrates that it is hard for attackers to compute  $T$  and  $T_2$  accurately to conceal their misbehaviors.

## 6 RELATED WORK

Cryptography provides networks with basic security services such as authentication, message integrity and

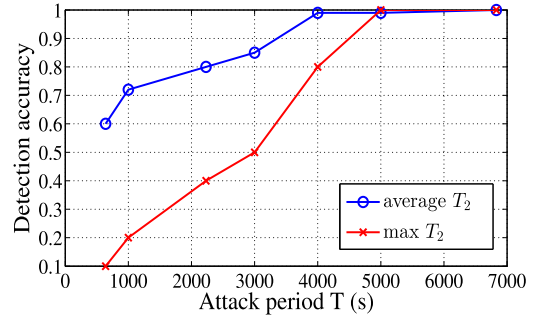


Fig. 19. Detection accuracy in behavior-switching attack.

non-repudiation. Several works [20], [21], [22], [23] have focused on designing cryptography schemes suitable in DTN and applying cryptography to the links to secure against malicious nodes. The cryptography approach can protect networks from unauthenticated external adversaries. However, it is not enough to defend against authenticated internal adversaries who launch such attacks as blackhole and greyhole attacks. Therefore, researchers have proposed misbehavior detection schemes to detect and mitigate insider attackers.

FBIDM [9] and MUTON [10] are proposed to detect blackhole in DTN using Prophet routing protocol [17]. Attackers utilize the insecurity of Prophet to arbitrarily increase its delivery predictabilities to absorb more packets. Trusted ferries collect the history of latest delivery predictability values of different nodes and cross-check the consistency of claims from different nodes. The methods only work for Prophet-based network. PMDS [11] involves a trust authority to probabilistically investigate a suspected node by collecting relevant secured evidences (contact, delegation and forwarding history) over all the network. The works [9], [10], [11] utilize the centralized server which might become the central point of attack and they are not suitable for distributed network.

Li and Das [12] suggests a distributed trust-based framework in which the forwarding behavior of a node is acknowledged by its next hop and a forwarding receipt is sent out to other nodes to update its reputation. Li et al. [13] is the first work to exploit contact evidences to defend against blackhole attack. Authenticated encounter records make it impossible for the adversary to claim non-existent encounters and abuse them to forge routing metrics to attract data. As the method cannot detect malicious packet-dropping, it is not sufficient to thwart the blackhole attack. In MDS [14], nodes exchange and use latest encounter records to estimate the forwarding ratios for others and assess them accordingly. The works [12], [13], [14] do not consider cooperating misbehaviors and cannot defend against collusion attacks.

Li's scheme [15] suggests that each contact record records the initial message buffer, sent and received messages during an encounter. A misbehaving node dropping packets received from the previous contacted node, will be detected by the next contacted node as it can observe the disappearance of packets in the message buffer records. However, Li's scheme produces high misjudgement rate of good nodes as it cannot differentiate the dropping behavior of malicious nodes and normal nodes.

## 7 CONCLUSION

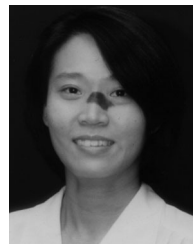
Blackhole and greyhole attacks are a serious threat to wireless networks, including DTNs. Existing solutions [9], [10], [11], [12], [13], [14], [15] can defend against packet dropping attacks in DTN but most of them fail to prevent the collusion of malicious nodes. We propose a method SDBG that can successfully prevent not only individual attackers but also cooperating attackers. The simulation results show that SDBG can detect colluding malicious nodes with high detection rate and low false positive rate when varying the number of colluding nodes and with a wide range of packet-dropping probability and different routing protocols.

## REFERENCES

- [1] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proc. Conf. Appl., Technol., Archit. Protocols Comput. Commun.*, 2003, pp. 27–34.
- [2] M. J. Khabbaz, C. M. Assi, and W. F. Fawaz, "Disruption-tolerant networking: A comprehensive survey on recent developments and persisting challenges," in *Proc. IEEE Commun. Surveys Tuts.*, 2012, vol. 14, pp. 607–640.
- [3] Delay-tolerant networking research group [Online]. Available: <http://www.dtnrg.org/wiki>
- [4] S. Burleigh, V. Cerf, R. Durst, K. Fall, A. Hooke, K. Scott, and H. Weiss, "The interplanetary internet: A communication infrastructure for Mars exploration," in *Proc. Committee Space Res. Sci. Assem.*, 2002, 365–37.
- [5] J. Partan, J. Kurose, and B. N. Levine, "A survey of practical issues in underwater networks," in *Proc. 1st ACM Int. workshop Underwater Netw.*, Sep. 2006, pp. 17–24.
- [6] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. S. Peh, and D. Rubenstein, "Energy-efficient computing for wildlife tracking: Design tradeoffs and early experiences with ZebraNet," in *Proc. 10th Int. Conf. Archit. Support Program. Lang. Oper. Syst.*, Dec. 2002, pp. 96–107.
- [7] S. Guo, M. H. Falaki, E. A. Oliver, S. Ur Rahman, A. Seth, M. A. Zaharia, and S. Keshav, "Very low-cost Internet access using KioskNet," in *Proc. ACM SIGCOMM Comput. Commun. Rev.*, Oct. 2007, vol. 37, no. 5, pp. 95–100.
- [8] J. Ott and D. Kutscher, "A disconnection-tolerant transport for drive-thru internet environments," in *Proc. IEEE 24th Annu. Joint Conf. Comput. Commun. Soc.*, Mar. 2005, pp. 1849–1862.
- [9] M. Chuah, P. Yang, and J. Han, "A ferry-based intrusion detection scheme for sparsely connected ad hoc networks," in *Proc. 4th Annu. Int. Conf. Workshop Security Emerging Ubiquitous Comput.*, 2007, pp. 1–8.
- [10] Y. Ren, M. Chuah, J. Yang, and Y. Chen, "MUTON: Detecting malicious nodes in disrupt-tolerant networks," in *Proc. IEEE Wireless Commun. Netw. Conf.*, 2010, pp. 1–6.
- [11] Z. Gao, H. Zhu, S. Du, C. Xiao, and R. Lu, "PMDS: A probabilistic misbehavior detection scheme toward efficient trust establishment in Delay-tolerant networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 22–32, Jan. 2014.
- [12] N. Li and S. K. Das, "A trust-based framework for data forwarding in opportunistic networks," *Elsevier J. Ad Hoc Netw.*, vol. 14, pp. 1497–1509, 2013.
- [13] F. Li, J. Wu, and A. Srinivasan, "Thwarting blackhole attacks in disrupt-tolerant networks using encounter tickets," in *Proc. INFOCOMM*, 2009, pp. 2428–2436.
- [14] Y. Guo, S. Schildt, and L. Wolf, "Detecting blackhole and greyhole attacks in vehicular delay tolerant networks," in *Proc. IEEE 5th Int. Conf. Commun. Syst. Netw.*, Jan. 2013, pp. 1–7.
- [15] Q. Li and G. Cao, "Mitigating routing misbehaviors in disruption tolerant networks," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 2, pp. 664–675, Apr. 2012.
- [16] A. Keranen, J. Ott, and T. Karkkainen, "The one simulator for dtn protocol evaluation," in *Proc. 2nd Int. Conf. Simul. Tools Tech.*, Rome, Italy, Mar. 2009, pp. 55:1–55:10.
- [17] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," *SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 7, pp. 19–20, 2003.
- [18] J. Burgess, B. Gallagher, D. Jensen, and B. N. Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networks," in *Proc. IEEE Int. Conf. Comput. Commun.*, Barcelona, Spain, Apr. 2006, pp. 1–11.
- [19] T. Spyropoulos, K. Psounis, and C. S. Raghavendra, "Spray and wait: An efficient routing scheme for intermittently connected mobile networks," in *Proc. ACM SIGCOMM Workshop Delay Tolerant Netw.*, Philadelphia, PA, USA, Aug. 2005, pp. 252–259.
- [20] N. Asokan, K. Kostianen, P. Ginzboorg, J. Ott, and C. Luo, "Towards securing disruption-tolerant networking," Nokia Research Center, Tech. Rep. NRC-TR-2007-007.
- [21] R. Patra, S. Surana, and S. Nadevski, "Hierarchical identity based cryptography for end-to-end security in DTNs," in *Proc. Intell. Comput. Commun. Process.*, 2008, pp. 223–230.
- [22] S. Roy, and M. Chuah, "Secure data retrieval based on ciphertext policy attribute-based encryption (CP-ABE) system for the DTNs," Lehigh University, USA, 2009.
- [23] Z. Jia, X. Lin, S. H. Tan, L. Li, and Y. Yang, "Public key distribution scheme for delay tolerant networks based on two-channel cryptography," *J. Netw. Comput. Appl.*, vol. 35, no. 3, pp. 905–913, 2012.



**Thi Ngoc Diep Pham** received the BS degree from Nanyang Technological University (NTU), Singapore in computer science in 2012 and is currently working toward the PhD degree in the School of Computer Engineering, Nanyang Technological University (NTU). Her research interests include security and privacy of mobile ad hoc networks, delay tolerant networks, and vehicular networks.



**Chai Kiat Yeo** received the BEng (Honors) and MSc degrees in 1987 and 1991, respectively, both in electrical engineering, from the National University of Singapore and the PhD degree from the School of Electrical and Electronics Engineering, Nanyang Technological University (NTU), Singapore, in 2007. She was a principal engineer with Singapore Technologies Electronics and Engineering Limited prior to joining NTU in 1993. She was the deputy director of the Centre for Multimedia and Network Technology (CeMNet) at Nanyang Technological University, Singapore, before her current appointment as the associate chair (academic) with the School of Computer Engineering, NTU. Her current research interests include ad hoc and mobile networks, delay tolerant networks, overlay networks, anomaly detection in cyber security, and speech processing and enhancement.

► For more information on this or any other computing topic, please visit our Digital Library at [www.computer.org/publications/dlib](http://www.computer.org/publications/dlib).