Detecting Blackhole Attacks in Disruption-Tolerant Networks through Packet Exchange Recording

Yanzhi Ren*, Mooi Choo Chuah†, Jie Yang*, Yingying Chen*
*Dept. of ECE, Stevens Institute of Technology † Dept. of CSE, Lehigh University
Castle Point on Hudson, Hoboken, NJ 07030 Bethlehem, PA 18015
{yren2, jyang, yingying.chen}@stevens.edu *chuah@cse.lehigh.edu*

***Abstract*—The Disruption Tolerant Networks (DTNs) are especially useful in providing mission critical services such as in emergency networks or battlefield scenarios. However, DTNs are vulnerable to insider attacks, in which the legitimate nodes are compromised and the adversary nodes launch blackhole attacks by dropping packets in the networks. The traditional approaches of securing routing protocols can not address such insider attacks in DTNs. In this paper, we propose a method to secure the history records of packet delivery information at each contact so that other nodes can detect insider attacks by analyzing these packet delivery records. We evaluated our approach through extensive simulations using both Random Way Point and Zebranet mobility models. Our results show that our method can detect insider attacks efficiently with high detection rate and low false positive rate.**

I. Introduction

Ad hoc networks can be easily deployed because they do not require fixed network infrastructures such as base stations or routers. Due to its self-organizing nature, an ad hoc network can be formed in real-time where all participating nodes perform packet forwarding. Thus, ad hoc networks are flexible and can provide mission critical services especially in emergency applications and battlefield scenarios.

However, in practice, due to the unstable paths caused by the high node mobility, low node density, and short radio ranges, traditional ad hoc routing protocols do not work well in DTNs. For instance, nodes are sparsely connected in tactical fields for the search or rescue missions. To address this issue, the Disruption Tolerant Network (DTN) concept [1] is introduced, which uses a store-and-forward approach to deal with such challenging network scenarios. In DTNs, nodes store packets if they cannot find a next-hop node to deliver them to destinations. The routing protocols in DTNs, e.g., MaxProp [2] and Prophet [3], require each node first stores packets in its memory and then selectively transmits packets when it encounters other nodes based on various metrics including the last encounter time, the numbers of previous encounters, and the estimated packet delivery probability values to other nodes.

Such metrics are derived from information provided by forwarding nodes themselves and it is hard to verify due to the network sparseness as well as the intermittent connectivity between nodes. Further, the portability of modern devices makes them tempting targets for thefts. Moreover, authenticated devices in chaotic battlefield environments are also likely to be captured by the enemy. Thus, it is easy for an adversary to compromise nodes within the network and launch insider attacks using the compromised nodes. Insider attacks can cause significant problems in networks. For instance, the adversary can use the compromised nodes to launch a particular harmful attack called blackhole attack. In a blackhole attack, the compromised nodes can become malicious by partially or entirely dropping data packets while participating correctly in the routing process. Further, the malicious nodes can use faked packet delivery probability to increase their chances of being selected as the next hop nodes and attract more data packets from other nearby nodes.

Most of the current work focus on securing routing protocols [4]. However, they cannot address insider attacks launched by compromised nodes. In [5], each node acts as monitoring nodes and overhears its neighbors' traffic to detect the anomaly in the network. However, this method may not be practical in DTNs because there are not enough monitoring nodes in the sparse network. Recently, the concept of encounter tickets is introduced in [6] to secure the evidence of each contact. However, this method can only prevent the attacker from claiming non-existent encounters and cannot cope with the packet dropping in the blackhole attack.

Recent work [7], [8] on detecting blackhole attacks rely on the introduction of a trusted examiner called ferry node, which moves around in the network and validates the packet delivery probability to determine the presence of the blackhole attack. In this paper, without relying on a third-party ferry node, we introduce a scheme that generates the un-forgeable packet delivery records in each contact and exploits the history of the packet delivery records to perform blackhole attack detection. In particular, when two nodes encounter each other, they will record the number of packets exchanged between them, and generate the secure records for each other with their private keys. In our scheme, when a node reveals its history packet records to its neighboring nodes, these nodes perform check and analyze the records to decide the sanity of this node.

To evaluate the effectiveness of our method, we conducted simulation experiments using a 40-node network where the nodes run a representative DTN routing protool PROPHET. We tested with different node densities and different movement patterns. From the results of simulations, we demonstrate that our method can detect the blackhole attack effectively and efficiently.

The rest of the paper is organized as follows. Section II discusses the related research. In Section III, we introduce the attack model that we considered in our work. In Section IV, we describe our detection scheme. We describe our simulation methodology with an overview of the PROPHET routing

978-1-4244-7265-9/10/$26.00

protocol, and present our simulation results in Section V. Finally, we conclude our work in Section VI.

II. Related Work

There are studies that address attacks launched from compromised nodes within a network. [9] attempted to address the survivability problem of the routing service when selective dropping attacks were launched. They used trusted nodes to monitor their neighbors. However, this method did not work well in DTNs where the network can be so sparse that there were not enough neighbors to act as the monitoring nodes. [5] required that each node overheard all traffic of its neighbors and then compared the values it observed with some metrics to detect the abnormal behaviors. This kind of approach required nodes to be in promiscuous mode and process all overheard packets, and thus it can be energy consuming. Futher, in sparse networks nodes might not hear its neighbor's transmission due to insufficient transmission power.

A ferry based detection method (FBIDM) [7] is proposed to detect malicious nodes and mitigate blackhole attacks by introducing a trusted examiner, ferry node. However, they did not consider the transitive property when calculating the delivery probability which is an important property reflecting the encountering of nodes in DTNs. Recently, an improved ferry based detection method called MUTON [8] is proposed. In this work, the transitive property is considered and MUTON can achieve better detection performance than FBIDM. However, MUTON still uses trusted ferry nodes in its detection mechanism and thus additional devices need to be deployed in the network which may not be economical or feasible.

The work that is most closely related to ours is [6], which introduced the concept of encounter tickets to secure the evidence of each contact. In [6], the nodes adopt a unique way of interpreting the contact history by making observations based on the encounter tickets. However, this method can only prevent the attacker from claiming non-existent encounters and can not cope with the packet dropping in the blackhole attack. Our work is different from [6] in that our method secures the packet delivery records instead of encounter records to detect the blackhole attack. Because the packet dropping is the key threat of blackhole attacks, our method can be effective to multiple blackhole attack scenarios. Further, our scheme can also detect deletion of packet delivery records, which is one of the further possible malicious behaviors.

III. Attack Model

We consider the following attack model in this work: a number of nodes in the network are compromised by adversaries. Once compromised, these nodes will maliciously affect the normal data delivery process in the network by launching blackhole attacks.

In particular, a compromised node will attract more packets by faking packet delivery probability to other nodes, and then drop some of these packets instead of forwarding them. The metric delivery probability $P(A, B)$ indicates how likely it is that the node A could deliver a packet to the destination node B and a node will also forward the data to another node it encounters if that node has a higher delivery probability to the destination than itself. In our attack model, the compromised node declares a higher random packet delivery probability to other nodes, which is larger than a threshold. For instance, to attack node B, the compromised node A can randomly choose a delivery probability of $P(A, B)$ greater than the threshold and advertise this fake $P(A, B)$ value to other nodes. Such actions will increase the chance of a compromised node being selected as the next hop node for relaying packets to the nodes that are being attacked (e.g., node B).Once selected as the next hop node, the compromised node will drop certain percentage (e.g., 50%) of the data it receives from other nodes and undermine the normal data delivery process in the network.

IV. Detection Method

In this section, we first provide an overview of our detection scheme. We then describe how the packet recording is performed at each node, which is the key component in our scheme. Further, we present the flow of the the detection process and theoretically analyze the width of the history window for packet recording.

A. Overview:

We assume each node in the network is issued a private key (RK) and a public key (PK) pair. Further, each node possesses other nodes' public keys. This is a reasonable assumption as it can be easily achieved by manually pre-loading all keys into the nodes during a network setup phase [10] or using a key distribution scheme, which is based on the mobility of the nodes and node encounters [6]. The private key at each node will be used to generate the un-forgeable packet receiving and forwarding information at each node. Each node keeps its packet receiving and forwarding records, created by the nodes which it encounters, in its memory. When two nodes meet, they will validate the history records from each other, determine the sanity of the encoutnering node and further detect the presence of the blackhole attack launched by the encoutnering node.

B. Recording Packet Exchange:

When two nodes encounter each other, they will exchange packets and perform recording of the packet exchange information. We use node A and node B as an example to illustrate how this recording process is carried out in our scheme. Node B generates the packet record for node A as follows: it includes the IDs of node B and node A, the number of receiving packets from node A, the number of forwarding packets to node A and the current time-stamp t in the record. Then node B signs the record using its private key. The format of the record generated by node B is as follows:

$$record{=}A,B,t,N_{rec},N_{send},E_{RK_B}(\mathbf{H}(A|B|t|N_{rec}|N_{send})). \quad (1)$$

Here $\mathbf{H}(*)$ denotes the hash function, $A|B|t|N_{rec}|N_{send}$ denotes the concatenation of node A's ID, node B's ID, encountering timestamp t, the number of receiving packets from node B $N_{receive}$ and the number of forwarding packets to node B N_{send}. The $E_{RK_B}(*)$ denotes the encryption using

node B's private key. The record includes the signature of node B and thus it can prevent fabrication or modification. Without B's private key, other nodes can not forge the record. After the record is generated, node B will send the record to node A. And node A stores this information in its memory. Similarly, node A will generate the record for node B to store in its memory.

Particularly, there are two tables generated at each node for storing such records, *Receiving Record Table (RRT)* and *Self Record Table(SRT)*.

Receiving Record Table (RRT): In this table, a node keeps packet exchange records generated by its encountering nodes. To save memory, each node only keeps the records generated during the most recent history window interval. Apparently, there is a tradeoff between the width of the history window and the performance of the attack detection. We discuss the choice of the history window in Section IV-D. detection. During a node encounter, upon receiving the request message, the node will send this table to its encountering node for node sanity check.

Self Record Table (SRT): In this table, a node maintains the records it generates for each node encounter. Each record contains the information of the encountering node ID and the time when the node encounter happens. For example, when node B encounters with node A, it generates a record for node A to store in A's RRT. Meanwhile, node B also generates a corresponding record containing node A's ID and the encountering time, and stores it in its own SRT: $record = A, t$. Similar to RRT, SRT only stores the records for the most recent history window. During the sanity checking process at each node encounter, the node will use the records in SRT to determine whether the encountered node has dropped the records generated by this node previously.

C. Detection Flow:

As an example, we illustrate our detection flow by showing how node A performs sanity checking of node B during the encounter of node A and B. The flow chart of our proposed method is depicted in Figure 1.

Step 1: When node A encounters node B, node A requests node B's RRT. After receiving node B's RRT, node A can determine the packet exchange information between node B and other nodes during previous encounters in the history window T. Node A then calculates the packet forwarding percentage of node B ψ, which is the ratio between the total number of packets that are sent out by node B during the history window T and the total number of packets recieved by B, for which B should forward in the history window T.

Assuming that there are M encounter events in the history window T, i.e., E_0, E_1, E_2,...,E_M, during which node B encounters with node j_0, j_1, j_2, ..., j_M. The packet forwarding percentage ψ can then be expressed as:

$$\begin{aligned}\psi &= \frac{actual\ forwarded\ packets}{packets\ need\ to\ be\ forwarded} \\ &= \frac{N_{send}^{E_0} + N_{send}^{E_1} + N_{send}^{E_2} + ...N_{send}^{E_M}}{N_{rec}^{E_0} + N_{rec}^{E_1} + N_{rec}^{E_2} + ... + N_{rec}^{E_M}}.\end{aligned} \tag{2}$$

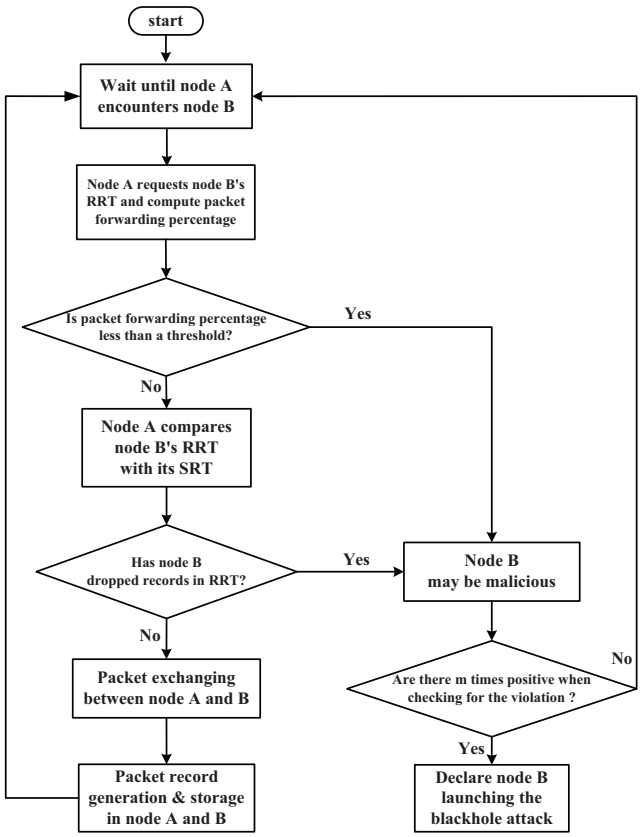


Fig. 1. The detection flow of the presence of a blackhole attack by using node A performs sanity checking of node B as an example.

$N_{send}^{E_0}$, $N_{send}^{E_1}$, $N_{send}^{E_2}$, ..., $N_{send}^{E_M}$ are the number of packets node B sent out during the previous encounter events E_0, E_1, E_2, ..., E_M respectively. The $N_{rec}^{E_0}$, $N_{rec}^{E_1}$, $N_{re}^{E_2}$, ..., $N_{rec}^{E_M}$ are the number of packets node B received during the encounter events E_0, E_1, E_2, ..., E_M respectively. if ψ is less than a threshold, it indicates that node B may selectively drop packets and will be listed as a suspicious node launching a blackhole attack.

Step 2: If node B is a malicious node, it is possible that node B removes some of the records that other nodes sent to be stored in its RRT, even though B cannot modify the contents of any record. Node A detects this misbehavior of node B by comparing the RRT with its own SRT. If some records in node A's SRT do not have corresponding entries in node B's RRT, node A will conclude that node B has dropped some records and list it as a suspicious node.

Step 3: If node B passes the sanity checking from node A and vice versa, node A and node B will exchange packets. They then generate a packet exchange record based on the description in Section IV-B and send it to store in each other's RRT. Further, both nodes will generate the corresponding record to store at their own SRT.

Step 4: If node B has been listed as a suspicious node more than m times (e.g., three times) by node A, node A will then declare node B as a malicious node that is performing the blackhole attack.

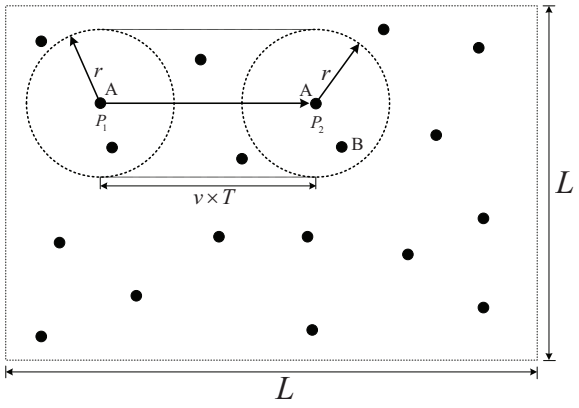


Fig. 2. Illustration of the probability that node A encounters with node B in the history window T

D. Choice of History Window:

The choice of the history window size T is a trade off between the detection performance and memory usage. We develop a method on how to choose a suitable T value based on the number of encounters N_{enc}.

We assume that N nodes are placed uniformly in a L by L meters square area as shown in Figure 2. The transmission range of each node is r meters and the average speed of the nodes is set as v. Node A moves from Position P_1 to Position P_2 with a speed of v during the time interval T.

From Figure 2 we can see that the probability that a node A encounters with another node B in the history window T is:

$$p = \frac{v \times T \times 2r + \pi r^2}{L \times L}. \tag{3}$$

If N_{max} is the maximum number of nodes that node A may meet during T, then the probability that node A encounters with exactly k nodes during time width T follows binomial distribution:

$$p(k) = \binom{N_{max}}{k} \times p^k \times (1-p)^{N_{max}-k} \tag{4}$$

Thus, the probability that node A encounters with at least N_{enc} nodes in the history window T is:

$$p(k \geq N_{enc}) = \sum_{k=N_{enc}}^{N_{max}} p(k) > \tau \tag{5}$$

For a given number of encounters N_{enc} and a fixed threshold τ, we can compute the width of the history window T from Equation (5). Alternatively, given a history window T value, one can estimate the number of encounters N_{enc} using Equation (5). For example, if we choose $N_{enc} = 5$, and set $N_{max} = 40$, $\tau = 0.8$, $L = 3000$ meters, $v = 3$ meters/second and $r = 250$ meters, then we can compute the needed history window T should be about 1050 seconds from Equation (5).

V. Simulation Evaluation

In this section, we first provide an overview of the PROPHET routing protocol, which we used in our simulations to evaluate our detection scheme. We then describe the simulation methodology and metrics. Finally, we present the simulation results of detecting malicious nodes.

A. Overview of PROPHET

PROPHET [3] is a routing protocol proposed for DTNs, which uses history of node encounters and transitivity. In PROPHET, three equations are used to update the delivery probability values: The node A will update its metric whenever it encounters with another node B using Equation (6):

$$P(A,B) = P(A,B)_{old} + (1 - P(A,B)_{old}) \times \alpha. \tag{6}$$

where α is an initialization constant which is set to 0.75. If a pair of nodes A and B do not encounter each other for a time period, node A would update its delivery probability to node B using Equation (7):

$$P(A,B) = P(A,B)_{old} \times \gamma^k \tag{7}$$

where γ is the aging constant which is set to 0.98. In addition, the delivery probability also has a transitive property: when node A encounters node B, which encountered node C previously, node A will update its delivery probability to node C based on the delivery probabilities of $P(A,C)$ and $P(B,C)$ using Equation 8:

$$P(A,C) = P(A,C)_{old} + (1 - P(A,C)_{old}) \times P(A,B) \times P(B,C) \times \beta \tag{8}$$

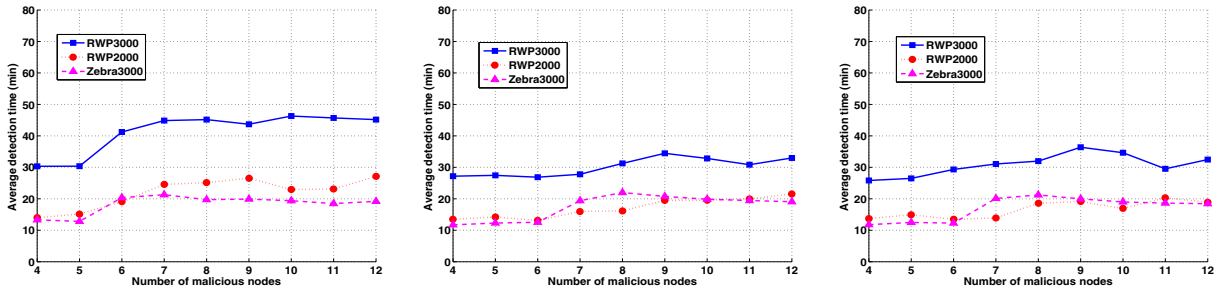
where β is a scaling constant that controls the impact the transitivity value has on the delivery predictability and is set to 0.25. The PROPHET routing protocol is used in our simulation to evaluate the performance of MUTON.

B. Simulation Methodology

1) Simulation setup: To evaluate the effectiveness of our proposed scheme, we conducted simulations by using the network simulator NS-2. In our simulation, we randomly deployed 40 nodes in a square area. The transmission range of each node is set to be 250 meters. During each experimental run, we randomly selected 10 pairs of nodes attempt to communicate with each other and set a constant bit rate (CBR) connection, 1 packet per second, for each pair. The packet size is 512 bytes and the buffer size at each node is 600 packets.

In our simulation, we studied two different movement patterns, the random way point model (RWP) and the Zebranet mobility model [11]. For the RWP model, the maximum speed is set to 5 meters per second with the pause time of 10 seconds. For each simulation run, the CBR connection will generate data in the first 3000 seconds and the simulation time is set to be 10000 seconds. We set the first 1000 seconds as the warm up time. We varied the width of the history window T by using 1000, 2000 and 3000 seconds respectively in the simulation.

To evaluate the performance of our method under different node densities and different movement patterns of the nodes, we tested using the following three simulation scenarios: (1) 40 nodes randomly deployed in a 3000 by 3000 meters square area with the RWP mobility model; (2) 40 nodes randomly deployed in a 2000 by 2000 meters square area with the RWP mobility model; and (3) 40 nodes randomly deployed in a 3000 by 3000 meters square area with the Zebranet mobility model. For each scenario, we studied the impact of the percentage of compromised nodes in the network by varying the number of compromised nodes from 4 to 12 among those 40 nodes. The compromised nodes start to conduct attacks in the network



(a) $T = 1000$ seconds (b) $T = 2000$ seconds (c) $T = 3000$ seconds

Fig. 3. Detection time versus number of malicious nodes under different history windows T.

after the warm up period. The simulation results presented for each scenario are the average results of 50 experimental runs.

2) Metrics: We use the following metrics to evaluate the efficiency and effectiveness of the detection schemes: (a) **average detection time:** it measures the average time for successfully detecting all the malicious nodes in the network. This metric measures the efficiency of the detection scheme; (b) **false positive rate:** it is the percentage of legitimate nodes that are mistakenly detected as the malicious nodes; and (c) **detection ratio:** it is defined as the percentage of malicious nodes that are detected by detection schemes. The false positive rate and the detection ratio show the effectiveness of the detection scheme.

C. Simulation Results and Analysis

1) The efficiency of the scheme: Figure 3 presents the simulation results of the average detection time versus number of malicious nodes under three different width of history windows T. In Figure 3 (a), the width of the history window is set as 1000 seconds. We observed that the average detection time of malicious node in scenario 1 (i.e. RWP3000) is between 30 minutes and 50 minutes as the number of malicious nodes increases from 4 to 12, whereas it is between 10 minutes and 30 minutes for scenarios 2 (i.e. RWP2000) and 3 (i.e. Zebranet3000). The detection time in scenarios 2 and 3 is about 20 minutes shorter than that in scenario 1. This is because the probability that one node encounters with other nodes increases with an increasing node density. Thus, more packets will be delivered and the detection time decreases in scenario 2. In scenario 3, the Zebranet mobility model is used: the nodes move faster and their movements are more chaotic. Therefore, there are more encounter events between nodes and more packets will be exchanged. Thus, it takes a shorter time to detect the blackhole attacks in scenario 3.

With the increasing width of the history window in Figure 3 (b), we found that the average detection time in scenario 1 is between 28 minutes and 35 minutes as the number of malicious nodes increases from 4 to 12, while the detection time of scenarios 2 and 3 is between 10 minutes and 20 minutes, which is also shorter than that of scenario 1. Compared to the detection time in Figure 3 (a), the results are shorter when the width of the history window increases to 2000 seconds. This is because for both RWP and Zebranet mobility models, with a longer history window T, the nodes can utilize more sending and receiving packets to perform detection, consequently, the blackhole attack can be detected faster.

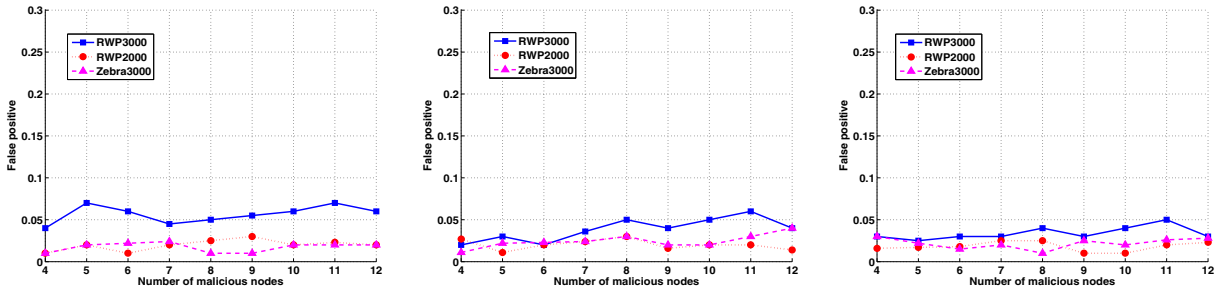
Furthermore, Figure 3 (c) presents the simulation results when the width of the history window is 3000 seconds. We observed that the average detection time remains stable as the width of the history window increases from 2000 seconds to 3000 seconds. This indicates that history window of 2000 seconds is long enough to achieve a stable average detection time.

2) The effectiveness of the scheme: We further study the effectiveness of our method in terms of detection ratio and false positive rate.

Figure 4 presents the simulation results of false positive rate versus the number of malicious nodes under different widths of history window T. From Figure 4 (a), we first observed that our method can achieve a low false positive rate across three scenarios: the false positive rate is around 5% in scenario 1, whereas it is about 3% in scenarios 2 and 3. Second, the false positive rate in scenarios 2 and 3 is lower than that in scenario 1. The reason is that the probability that one node encounters with the other nodes is higher with denser node density. Therefore, more packets will be delivered as the node density increases, and thus the attack detection can become more accurate and the false positive rate decreases in scenario 2. In scenario 3, the nodes move faster and more chaotic under the Zebranet mobility model, compared with the RWP model. Therefore, more encounter events exist between nodes and more packets can be exchanged. Thus, compared with scenario 1, our scheme can detect the blackhole attack with lower false positive rate in scenario 3.

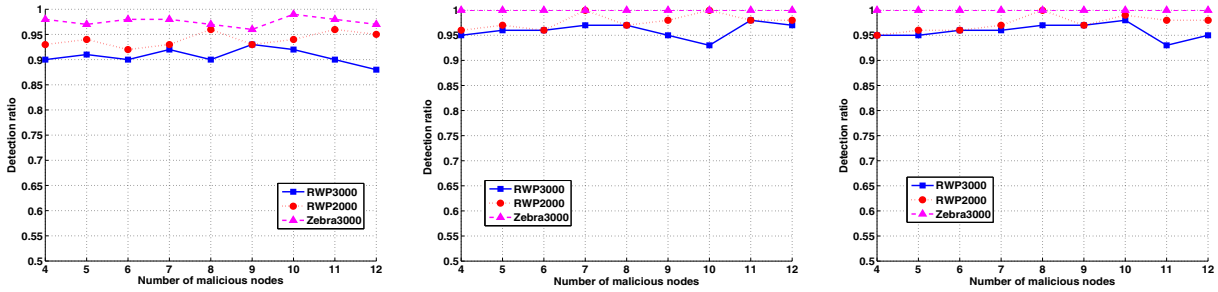
Figure 4 (b) shows the false positive rate when the width of history window increases to 2000 seconds. Similar to Figure 4 (a), the false positive rate is higher in scenario 1 than that of scenarios 2 and 3. Specifically, it is around 4% in scenario 1 and 2% in scenario 2 and 3 as the number of malicious nodes grows from 4 to 12. In addition, we observed that the overall false positive rate is lower when the history window is 2000 seconds. This shows that in a scenario with longer history window T, the lower false positive rate can be achieved because more packets can be used for blackhole attack detection and thus the detection result is more accurate.

Figure 4 (c) shows the simulation results for scenario 1 to 3 when the width of history window is 3000 seconds. Similar to the previous observations, we found that the false positive rate of scenario 1 is higher than that of scenario 2 and 3. Specifically, it is around 4% in scenario 1 and 2% in scenarios 2 and 3 as the number of malicious nodes grow from 4 to 12. In addition, we observed that the overall false



(a) $T = 1000$ seconds (b) $T = 2000$ seconds (c) $T = 3000$ seconds

Fig. 4. False positive rate versus number of malicious nodes using different widths for the time window T.



(a) $T = 1000$ seconds (b) $T = 2000$ seconds (c) $T = 3000$ seconds

Fig. 5. Detection ratio versus number of malicious nodes under different width of time window T.

positive rate in scenario 3 is comparable to that of scenario 2. It also showed that when the width of history window is large enough, the performance of false positve rate does not improve. In summary, the false positive rate can not be reduced by continuously increasing the width of the history window.

Finally, we studied the detection ratio versus the number of malicious nodes when the width of the history window T is 1000, 2000 and 3000 seconds, respectively. When the width of the history window is 1000 seconds, from Figure 5 (a), we found that the detection ratio is around 90% in scenario 1, whereas it is around 92% for scenario 2 and 97% for scenario 3, respectively. Higher detection ratio can be achieved if the network becomes denser or the nodes with higher mobility speed. In Figure 5 (b), we observed that the detection ratio is around 95% in scenario 1 and it is around 97% in scenario 2. In scenario 3, when the Zebranet mobility model is used, we observed that our method can achieve 100% detection rate. This is encouraging, as it indicates that our method can detect all the malicious nodes for various number of malicious nodes. When the width of history window increase to 3000 seconds, similar results are displayed in Figure 5 (c): the detection ratio of scenario 1 is around 95%, and it is around 96% in scenario 2. The detection ratio in scenario 3 is also 100%. Moreover, Figure 5 shows that our detection method achieves a stable performance with 2000 seconds history window, which is inline with the observations made for the average detection time and false positive rate. Overall, the performance of our detection scheme remains stable when the width of the history window reaches 2000 seconds.

VI. Conclusion

To detect the blackhole attacks launched by compromised nodes in Disruption-Tolerant Networks (DTN), we proposed a scheme that exploits the secure records of packet delivery information in each contact to detect the presence of the blackhole attack. The attack detection is performed based on the analysis of these un-forgable packet delivery records. Our scheme does not rely on a third-party trusted examiner and is easy to apply. We show that our method can effectively detect malicious nodes and mitigate the negative impact caused by the blackhole attack on the data delivery process in DTNs. Our extensive simulation under different mobility models demonstrates that our method can detect blackhole attacks effectively with high detection rate and low false positive rate.

References

[1] S. Farrell and V.Cahill, *Delay and Disruption Tolerant Networking*. Artech House, 2006.

[2] J.Burgess, B.Gallagher, D.Jensen, and B.N.Levine, "Maxprop: Routing for vehicle-based disruption-tolerant networking," in *Proceedings of IEEE Infocom*, April 2006.

[3] A. Lindgren, A. Doria, and O. Schelen, "Probabilistic routing in intermittently connected networks," in *Mobile Computing and Communications Review*, 2003.

[4] A. Perrig, R. Canetti, D. Song, and J. D. Tygar, "Efficient and secure source authentication for multicast," in *Network and Distributed System Security(NDSS) Syposium*, Feb 2001.

[5] Y.Huang and W.Lee, "A cooperative intrusion detection system for ad hoc networks," in *Proceedings of 1st ACM Workshop on Security of Ad Hoc Networks*, 2003.

[6] F. Li, J. Wu, and A. Srinivasan, "Thwarting black hole attacks in distruption-tolerant networks using encounter tickets," in *Infocom 2009*, 2009.

[7] M. Chuah, P. Yang, and J. Han, "A ferry-based intrusion detection scheme for sparsely connected adhoc networks," in *Proceedings of first workshop on security for emerging ubiquotous computing*, 2007.

[8] Y. Ren, M. C. Chuah, J. Yang, and Y. Chen, "Muton: Detecting malicious nodes in disruption-tolerant networks," in *WCNC 2010*, 2010.

[9] S. Marti, T.J.Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *6th ACM International Conference in Mobile Computing and Networks*, Augest 2000.

[10] B. Schneier, *Applied Cryptography*. John Wiley & Sons, 1996.

[11] Y. Wang, S. Jain, M. Martonosi, and K. Fall, "Erasure coding based routing for opportunistic networks," in *Proceeding of Sigcomm WDTN Workshop*, Augest 2005.