

Trust strategy implementation in OppNets

Asma'a Ahmad¹ · Robin Doss¹ ·
Majeed Alajeely¹ · Sarab F. Al Rubeai²

Received: 1 December 2016 / Accepted: 15 July 2017 / Published online: 22 July 2017
© Springer-Verlag GmbH Austria 2017

Abstract With the natural characteristics of Opportunistic networks (OppNets) where delivery is delayed with frequent disconnections between mobile nodes in dynamically changing routes to destinations, malicious nodes can perform selective packet dropping attacks easily without been identified easily. This is why securing the data flow without any loss becomes challenging in OppNets. In this paper, we present a solid trust based node and path detection technique against selective packet dropping attacks. Using the trust attribute with the Merkle hashing technique, a node's identity can be validated, and malicious nodes can be detected. We integrate our proposed technique with Epidemic routing and use simulation to show how effective the technique works against selective packet dropping attacks. We use simulation to show how the node detection accuracy increases with time, as intermediate nodes have more time to establish trust with destination nodes. We also use simulation to show that delivery rates increase with increased storage, and show how our trust model improves and secures routing compared to non-trust models.

Keywords Opportunistic networks · OppNets · Trust · Selective packet dropping attack · Merkle tree · Malicious node detection

Mathematics Subject Classification 60A10

✉ Asma'a Ahmad
asmaa.ahmad@deakin.edu.au

¹ Center for Research and School of Information Technology, Deakin University, Geelong, Australia

² Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada

1 Introduction and motivation

1.1 Introduction

In an opportunistic network (OppNet), when a source node needs to send a message to a destination node, it forwards the message to the destination node through nodes it meets with opportunistically. Having evolved from Delay Tolerant Networks (DTNs), the connection between nodes is not constant and the path to the destination node is not known. A node is chosen to deliver the message to the destination based on its ability and history to deliver the message competently. However, delays are normal, and message delivery is not guaranteed [1]. What differentiates OppNets from DTNs, is that there is never a fixed path to the destination node to forward the message. So the message forwarding is always opportunistic, and is based on the availability and ability of the node to forward the message to the destination. In DTNs however, a search for an existing path to the destination is made first. If the search gives no results, then an opportunistic approach is used to forward the message to the destination.

1.2 Motivation

Sometimes, the use of OppNets becomes mandatory when traditional networking is exposed to failures resulting from natural catastrophes or unexpected emergencies. In such unusual conditions, exchanging essential messages can only be done through an OppNet [2]. Some of the growing uses of OppNets include their use in Zebanets [3], a sensor network with tracking sensors attached to the animals (zebras). OppNets are also used in the military [4], in inter-planetary networks [5], and for natural disaster recoveries where opportunistic communication between people becomes the only option.

Selective packet dropping attacks cause a threat to OppNets because they breach the integrity and completeness of the message. To elaborate, a message is composed of a number of packets and the attack aims for dropping some of these packets as they are transferred from node to node towards their destination. Arriving at the destination with missing packets results in an altered message. This type of attack affects the integrity of the message once it reaches the destination and results in the degradation of network performance and can sometimes completely disrupt communication in the network [6]. Preserving the integrity of the message from the source to the destination nodes is very important in OppNets, yet a challenging task at the same time.

Existing non trust/reputation based techniques used to deal with packet dropping attacks [7–10] have limitations, some of them focus on redirecting or replicating messages, or detecting the malicious path but do not detect malicious nodes. Some methods aim for malicious behaviour or node detection but with a very high network cost and decreased delivery rates. To deal with attack detection, and deal with different types of malicious behaviour more efficiently, researchers have invested in developing security methods that are based on the trust and reputation attributes. The two attributes are essential to validate a node's identity and avoid malicious nodes. Trust and reputation

are used in P2P networks [11], ad hoc networks [12], MANETs [13], WSNs [14], DTNs [15], and OppNets [16].

1.3 Overview of solution

We define trust as how much confidence a node has towards another node and is used as a guide for nodes to make their routing decisions. For instance, assume two nodes—*A* and *B*—opportunistically met in a network. Node *A* needs to forward a message to the destination, so node *A* has the choice to decide whether it will trust node *B* to pass the message to it or not. Node *A* will make its decision based on how much confidence it has towards node *B*, or how much it trusts node *B*. The level of trust node *A* has towards node *B* defines the level of honesty and the good service that node *A* believes node *B* will provide. So, in our work, trust defines the level of honesty and the good service a node has in the network, and this is derived and built from direct past interactions a node has with others.

Trust is used to enhance and improve security in a network, however, at the start of the network, it is hard for nodes to trust each other knowing that they have never encountered each other before. In our work, we propose a dynamic trust model where over time the trust ratings of nodes are varied as nodes move, meet, and interact with each other in the network.

In this paper, we present a malicious path and a malicious node detection mechanism that detects malicious paths with nodes that attempt selective packet dropping attacks using the Merkle tree hashing technique [17]. Based on the path detection (whether legitimate or malicious) a trust value is built for nodes in each path. The built trust value for each node in each path the destination is involved with is stored locally in each destination's trust table. Malicious nodes are detected when their trust values fall below the required threshold. The advantage of using trust to identify malicious nodes is that nodes have more than one chance to prove themselves legitimate before they are listed as malicious.

Contribution The main contributions of this paper are:

1. Efficient algorithms to detect a malicious path using the Merkle tree hashing technique, where the Merkle root will be used to detect if a selective packet dropping attack has occurred.
2. An efficient algorithm that builds a trust value for nodes as they forward complete packets to destination nodes. Destination nodes create local trust tables for nodes with their trust values and are then used by destination nodes to classify malicious nodes and non-malicious nodes.

The remainder of this paper is organized as follows. In Sect. 2, we present the related work. In Sect. 3, we present the proposed trust strategy. In Sect. 4, we present the mathematical analysis. In Sect. 5, we present the simulation results. In Sect. 6, we present extended results on the impact of mobility speed and storage on malicious node detection accuracy. In Sect. 7, we compare our trust based approach with a non-trust based approach. Finally, in Sect. 8, we present the conclusion with future work.

2 Related work

Various defense methods have been established to secure and improve routing in OppNets from selective packet dropping attacks, which are common attacks in an OppNet. Trust and reputation took a great role and attention by researchers to create trust relationships in OppNets. Trust and reputation are important to authenticate a node's identity and avoid malicious nodes. Trust is a very important feature that needs good management in networks.

Li et al. [18] used probability to estimate the trust value of nodes sending honest information to other nodes. The authors developed a model that utilized legitimate nodes in the network. Initially, each node in the network was assigned a fixed trust value. Nodes then started to exchange information. When a node received the correct information from a node, it would update the trust value of the node it received the information from. To confirm a trust value of a node, a node might request from a node information that it already had and can compare with, making this request a couple of times it could confirm that the node was indeed honest. Legitimate nodes trust values were exchanged in the network until the trust value of legitimate nodes was spread in the network.

Cho et al. [19] proposed a history based trust model for DTNs. The model examined the data that is received from nodes in addition to the nodes that passed the data. The stochastic Petri net technique was used to evaluate the model. Legitimate nodes used a group key to interact with each other, the key was managed by a trust authority. Trust was formed from direct and indirect experiences, and changed dynamically according to environmental changes. The model worked effectively in dealing with increased network overhead when selfish and malicious nodes started affecting the network. However, even though the model used trusted authorities for group management only, they are costly and hard to provide in large networks.

Nguy et al. [20] used context to enhance a Bayesian network trust model in ad hoc networks. To improve trust in a network, the authors specified that context was composed of a group of facilities that were publicly available to all nodes in the network. For a node to choose the most suitable utility, it used past records with a given utility provider, and then made its choice. Trust was defined in their model as the probability of a utility provider to satisfactorily provide a utility to the node that requested its utility. A Bayesian network was created for each utility's measured trust from direct and indirect trust. The authors used context information (such as the node, the utility, the service provided by the utility, and the date of the interaction) to make conditions that may affect the outcome of an interaction experience between two nodes. The Bayesian network sorted these experiences based on their creation date. The model does not consider the mobility or the density context of the network when calculating trust.

To safeguard the Quality of Service (QoS) of data availability, Bijon et al. [21] proposed a probability based model that adopted the Dempster–Shafer theory (reasoning with uncertainty) that competently collected recommendations from intermediate nodes and effectively discarded malicious ones. Trust values were assigned and stored in the intermediate nodes of a path, the recommendations were prioritized based on the trust values of nodes in a path. Recommendations from nodes with

a higher trust value were prioritized over recommendations from nodes with lower trust values. Recommendations from shorter distanced nodes were given more priority over longer distanced nodes. The model also enhanced the trust values of nodes by measuring their ability to develop their trust, a node was given the choice to whether or not it wished to trust another node regardless of its recommendation value. This technique might not give fair results in a network with randomly mobile nodes.

A social based dynamic trust model was proposed by Yao et al. [22]. As nodes moved in the network, they exchanged messages with nodes that had a higher social similarity with destination nodes. The more common features nodes had with each other, the more trust they had towards each other. The trust that was formed is dynamic though, which meant it would change when the behavior of a node changed, thus trust was formed as a result of social similarity as well as the routing behaviors of nodes. The proposed model achieved improved performance against different attacks by allowing packets to flow through trusted nodes only, and isolating malicious and selfish nodes. More work is required to preserve the privacy protection for users, as the model relied on the assumption that users always share their social features to help other nodes.

A communal reputation and an individual trust based model in a WSN was presented by Zia et al. [23]. The model built reputation from trust formed by feedback from nodes about each other. To build trust in the network the model used voting and implemented the watchdog mechanism [24] where each node monitored its neighbour. Each node issued a trust vote for other nodes, and recorded their trust vote in a trust table. Positive votes (resulting from successful message delivery) increased the trust value of a node, and negative votes decreased the trust value of a node. If a node's trust value fell below the required threshold, it would be notified and reported as malicious to other nodes to bring awareness regarding this malicious node. Once this awareness reached the cluster head through multiple nodes, it would isolate the malicious node from the cluster by informing the nodes in the cluster to abandon any messages from the reported malicious node. In addition to the trust table that each node maintained in the network, each node also maintained a reputation table that included the evaluated reputation values for all other nodes in the network. Each node built the reputation table from its own trust table and other node's trust tables which were broadcast occasionally in a cluster. Using monitoring nodes requires to have nodes transmit messages within the same transmission range, also the scheme consumes a large overhead on the network.

Jaimes et al. [25] proposed an anonymous reputation system for vehicular ad hoc networks. The message forwarding was opportunistic and was forwarded through intermediate nodes. Intermediate nodes verified nodes using a set of pseudonyms, and checked the reputation value of nodes before accepting or rejecting any messages. When the message reached the destination node, the message was verified using the signature and the public key obtained from the pseudonyms set. The destination reported the result of message verification to the reputation server that updated the reputation values of the vehicles that were involved in passing the message to the destination.

A reputation system was proposed by Dini et al. [26] where every node locally assigned a reputation value to nodes it interacted with and used this reputation value in the future by choosing nodes with a high reputation to forward its message to. The

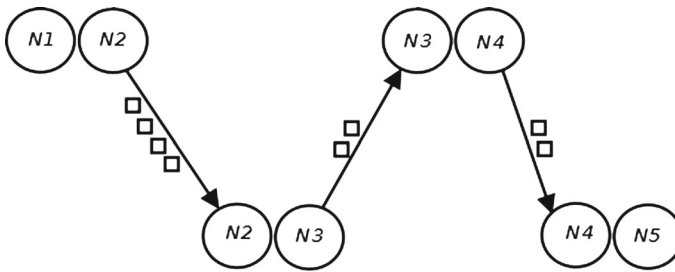


Fig. 1 A selective packet dropping attack

integrity of messages was protected using digital signatures. Each node maintained the reputation of other nodes using acknowledgments from the destination, a node list, and aging. When a source sent a message to a certain destination through intermediate nodes, the destination would send an acknowledgment back to the source once it received the message. When the acknowledgment arrived at the source; the source would update the reputation of intermediate nodes. Because the reputation system was built locally among nodes, this technique reduced the overhead cost of having to maintain a reputation system globally.

3 Trust strategy in OppNets

Selective packet dropping attacks are categorized under denial of service attacks because they breach the integrity of data flowing in the network, which affects the communication between nodes in the network and impacts its performance.

Selective packet dropping attacks don't always happen due to malicious reasons. Sometimes, packets are dropped by the node as a result of limited storage or loss or damage of the device. Sometimes the packet is dropped due to collisions or its life [(time to live (TTL)] has ended.

An example of a selective packet dropping attack is shown in Fig. 1. The path includes five nodes (represented by N_x). N_2 receives four packets from N_1 , when N_2 is in contact with N_3 , it forwards to N_3 the four packets it received from N_1 . After dropping two packets, when N_3 is in contact with N_4 it passes it only two packets. We clarify that the term 'selective' is used to indicate that not all packets in a message are dropped.

With the implementation of the developed algorithms, the path in Fig. 1 will be detected as malicious, and the trust values for the intermediate nodes N_2 , N_3 , and N_4 will be impacted negatively. When any node's trust value falls below the minimum threshold; it is then classified as malicious.

The technique developed in this paper to detect selective packet dropping attacks aims to achieve:

1. Malicious path detection using the Merkle tree technique.
2. Malicious node detection using trust.

3.1 Assumptions

The following are assumed:

- Intermediate nodes may be malicious but the source and destination nodes are assumed to be legitimate.
- Since we are considering selective packet dropping attacks, we assume that at least one legitimate packet reaches the destination.

3.2 Malicious path detection using Merkle trees

In OppNets, Merkle trees can be used to verify that data transferred in a path is complete. If one packet is removed, the hash of its parent will change. Which will change the hash of its parent, and so on, resulting in the Merkle root's hash to change as well.

A Merkle tree [17] is a special binary tree that starts with hashing every packet using a one-way hash function that takes an input and turns it into a unique hash using a mathematical algorithm. The resulting hashes of these packets are called the leaves of the tree ($[H_{00}]$, $[H_{01}]$, $[H_{02}]$, $[H_{03}]$, $[H_{04}]$, $[H_{05}]$, $[H_{06}]$, and $[H_{07}]$ in Fig. 2). Every pair of leaves are then concatenated and hashed to derive child hashes ($[H_{10}]$, $[H_{11}]$, $[H_{12}]$, and $[H_{13}]$ in Fig. 2). Depending on the size of the tree, the child hashes are then also concatenated and hashed to derive further child hashes ($[H_{20}]$, and $[H_{21}]$ in Fig. 2). This process is repeated until only one value is left, the root, or the Merkle root ($[H_{30}]$ in Fig. 2). The Merkle root can be used to authenticate messages, and can also reduce the authentication overhead needed.

Merkle trees are usually binary trees, and they are perfectly complete when the number of leaves/packets is even, and in each level of the tree (except the root), the number of hashes is also even. If the number at any level of the tree is odd, then we

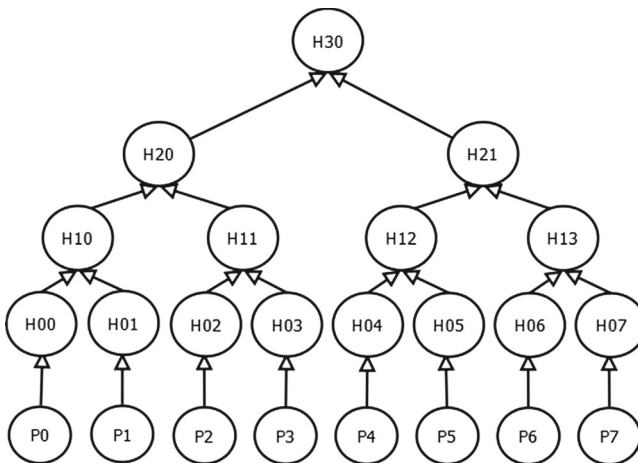


Fig. 2 A Merkle tree

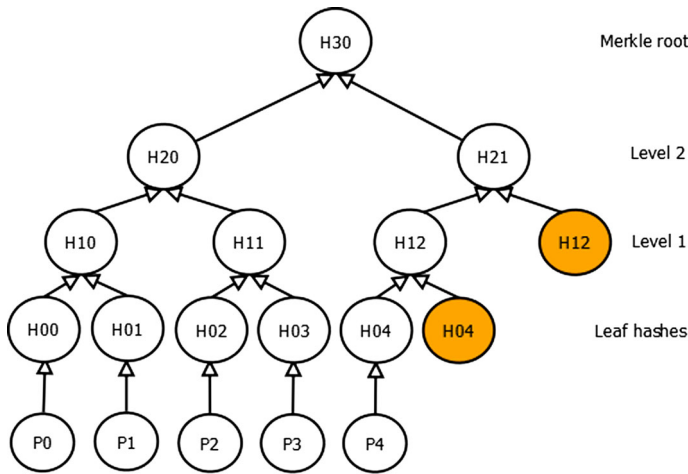


Fig. 3 A Merkle tree with number of hashes that needs completion

refer to the tree as a complete (but not perfect) tree. In our work, both of perfectly complete trees and trees that require completion are considered. At every level of the tree, if the number of hashes is not even, the odd hash is concatenated with itself to form a new parent hash value.

An example is shown in Fig. 3, the leaf hash value of the fifth packet is single (H_{04}), so it will be concatenated with itself to generate the parent hash value (H_{12}). At level 1, H_{12} is concatenated with itself to generate the parent hash value (H_{21}), and at level 2, H_{21} is merged with the rest of the tree. The concatenation process needs to happen at every level of the tree only when the resulting hashes in the level is not even. For example, in Fig. 3, the hashes needed self-concatenation at the “Leaf hashes” and “Level 1” levels, but not in “Level 2”.

Advantages of using Merkle trees include:

1. The advantage of using Merkle trees over cryptographic verifications is that packets can be verified using the root only first, if the received Merkle root matches with the calculated Merkle root, then there is no need to do further calculations, thus limiting the amount of data transferred which saves time and network resources [27].
2. They save computational resources needed to verify each packet. Hash calculations perform significantly fast because of their small size (about 100 or 200 bits long), thus they don't consume a lot of computational resources when compared to cryptographic verifications.
3. Sending and verifying all of the hashes individually is more expensive than the verification process we perform using Merkle trees where only the Merkle roots were verified.

In our technique, the sender (source node) performs algorithm 1, embeds the Merkle root in each packet's header and sends them to the destination through intermediate nodes, Fig. 4. Upon receipt, the destination node performs algorithm 1 to compute a new Merkle root.

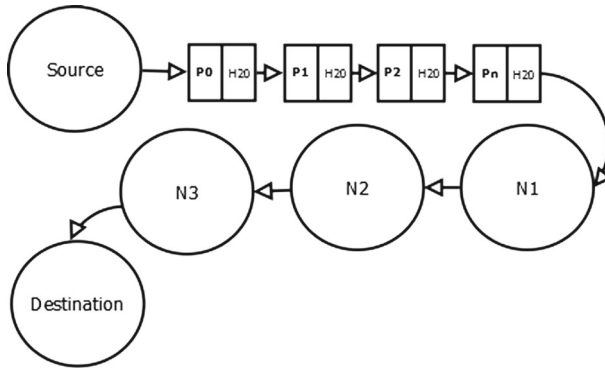


Fig. 4 Source sends the packets with embedded Merkle root to the destination

The reason why we choose to send the root in each packet's header, is to guarantee the arrival of the root even if only one packet is received.

In Algorithm 1:

- The source/destination hashes each packet to generate leaf hashes (lines 2–3).
- It builds a Merkle tree from the leaf hashes (lines 4–12).
- It computes the Merkle root (lines 13–15).

The source sends the packets along with the Merkle root inside each packet's header. The destination receives the packets; it performs Algorithm 1 to compute the hashes for the packets, builds a Merkle tree, and computes a new Merkle root.

Algorithm 1 Hash each packet, build Merkle tree, and compute the Merkle root

```

1: READ: packets
2: For all packets
3: hash[i] = createHash(packet[i])
4: For all hashes in each level
5: if numberOfHashes = even then
6:   hash[i] = createHash(hash[i] + hash[i])
7: else
8:   if !lastHash then
9:     hash[i] = createHash(hash[i] + hash[i + 1])
10:  else
11:    hash[i] = createHash(hash[i] + hash[i])
12:  end if
13:  if level = last then
14:    rootValue = hash[i]
15:  end if
16: end if
  
```

The destination then performs Algorithm 2 to use the computed Merkle root to compare with the original Merkle root sent by the source. If the 2 roots match, it means that the destination received the correct number of packets. If the 2 roots don't match, the destination reports the path as malicious, and reports that a selective

packet dropping attack has occurred. At this stage, the destination cannot identify the malicious node, and thus suspects that any node in the path could be malicious.

In Algorithm 2:

- If the 2 roots are not equal, the destination reports that the path is malicious, and the path is added to the malicious paths list (lines 2–6).

Original hash sent by the source is denoted with the symbol ($'$).

Algorithm 2 Identify malicious paths

```

1: READ: merkleRoot'
2: if merkleRoot  $\neq$  merkleRoot' then
3:   Path is malicious
4:   maliciousPaths[maliciousPathCounter] = currentPath
5:   maliciousPathCounter++
6: end if

```

3.3 Malicious node detection using trust

All nodes in the network are initially trusted in the network, and are assigned an initial trust value of 0.5. If a node's trust value decreases below 0.2 (nodes with trust values of 0.1 or 0.0), it is classified as malicious. Destination nodes perform the detection process based on the data they receive from source nodes through intermediate nodes.

As detection is done at the destination, the destination in this case acts as a central authority. Each destination maintains a trust table that records the direct trust value of each node that contributed in passing the message along a given path.

When the destination authenticates the message by matching the received Merkle root and the calculated Merkle root, it increments the trust value of each node in the path by x , where x could be any value in the range 0.1 and 0.4. The best value for x was identified through evaluations and happened to be 0.1. If the 2 roots don't match, the destination subtracts x from the trust value of each node in the path, as indicated in Algorithm 3. Because at this stage, the destination cannot identify the exact node that may have dropped the packets, and cannot know if they were dropped for malicious or non-malicious reasons. It subtracts x from the trust value of each node in the path and only counts a node as malicious if its trust value falls below 0.2 to give them an opportunity to prove in other paths that they are not malicious (if they are indeed legitimate). If a node's trust value falls below 0.2, and encounters the destination, the destination would still run Algorithms 1 and 2 because that node may have been falsely identified as malicious. The roots are still checked even if the node's trust value is below threshold. If the roots equal each other, the trust value of that node increases by x . This makes our trust mechanism dynamic to consider falsely identified nodes and non-malicious packet dropping reasons.

In Algorithm 3:

- If the 2 roots are equal, then no packets were dropped. The trust value for each node in the path that passed the message is increased by x (lines 2–4).

```

1 =====
2 PATH 1
3 =====
4 Nodes:          [S]-> [t24]-> [D]
5 Packets received:  [6]   [5]
6
7 Root value sent by source node: [a4c4f6eb2930a935142bfce94ca225815e92a273]
8 Root value created by destination node: [ba674f88a616c7590c435f33726a456126da6f06]
9
10 The 2 roots are different, the path is malicious
11 A selective packet dropping attack have occurred
12 -----
13 | Destination D |
14 -----
15 | Node |Trust value|
16 -----
17 | t24 | 0.4 |
18 -----

```

Fig. 5 Trust table maintained by destination node $[D]$ at path 1

- If the 2 roots are not equal, the trust value for each node in the path that passed the message is decreased by x (lines 5–7).

Algorithm 3 Build trust

```

1: READ: merkleRoot'
2: if merkleRoot = merkleRoot' then
3:   Packets are all legitimate
4:   trustValueForEachNode = trustValueForEachNode +  $x$ 
5: else
6:   trustValueForEachNode = trustValueForEachNode -  $x$ 
7: end if

```

Destination nodes utilize the trust tables they construct to identify good nodes from malicious ones. In this work, only destination nodes make use of the trust table. If the destination meets with a node that it did not work with before as an intermediate node in passing a message, then it assumes the default trust value assigned to it at the start of the network, which is 0.5.

For example, the trust table of destination node $[D]$ right at the start of the network is shown in Fig. 5, the first node it comes in contact with is node $[t_{24}]$, which happens to be malicious. Since the Merkle roots don't match, the destination node decreases the trust value of node $[t_{24}]$ by x , where x is 0.1.

The trust table of destination node $[D]$ at path 2 is shown in Fig. 6, when the Merkle roots didn't match, the destination node decreased the trust value of nodes $[t_{20}]$ and $[t_{24}]$. It cannot identify at this stage the malicious node, but using this trust mechanism, eventually, it will detect the malicious node later in the network.

4 Mathematical analysis

As indicated in Sect. 3, to attempt a selective packet dropping attack, the malicious node attempts to drop some but not all of the packets, thus, the assumption that at least one legitimate packet must reach the destination node was developed. Based on that,

```

19  PATH 2
20  =====
21  Nodes:      [S]-> [t20]-> [t24]-> [D]
22  Packets received: [6]   [5]   [5]
23
24  Root value sent by source node: [bd432db3c574b5d69ea07db694d7e45879512d4d]
25  Root value created by destination node: [ba674f88a616c7590c435f33726a456126da6f06]
26
27  The 2 roots are different, the path is malicious
28  A selective packet dropping attack have occurred
29  -----
30  | Destination D |
31  -----
32  | Node |Trust value|
33  -----
34  | t24 | 0.3 |
35  -----
36  | t20 | 0.4 |
37  -----
    
```

Fig. 6 Trust table maintained by destination node [D] at path 2

in this section we develop an expression for the probability of having a malicious path, resulting from a selective packet dropping attack with the condition that at least one legitimate packet reaches the destination. We assume that the dropping of the packets themselves is purely random. The notations used are as follows:

- n is the total number of nodes in the path
- m is the number of malicious nodes in the path
- k is the number of packets
- p is the probability that a packet is dropped by a malicious node
- β is the probability that at least one legitimate packet reaches the destination
- D_j is the probability of dropping j packets out of k packets

Since β is the probability that at least one legitimate packet reaches the destination. Thus:

$$\beta = 1 - \text{probability of no packets reaching the destination}$$

Given p , representing the probability that a packet may be dropped by a malicious node, therefore a packet to survive a malicious node is $1 - p$. This means that the probability for a packet to survive a malicious path of m nodes is $(1 - p)^m$. As a result, the probability of a packet to be dropped along a malicious path of m nodes is $1 - (1 - p)^m$, which is denoted by λ . Therefore, the probability for k packets to be all dropped in a path of m malicious nodes is $(\lambda)^k$, which means that the probability of no packets reaching the destination equals $(1 - (1 - p)^m)^k$. Hence the formula which represents the probability that at least one legitimate packet reaches the destination is,

$$\beta = 1 - (1 - (1 - p)^m)^k$$

From this formula, the minimum number of packets required to ensure packets delivery can be found to be:

$$1 - \beta = (1 - (1 - p)^m)^k$$

This means,

$$\log(1 - \beta) = k \log(1 - (1 - p)^m)$$

Therefore,

$$k = \frac{\log(1 - \beta)}{\log(1 - (1 - p)^m)}$$

Now the probability of dropping j packets out of k packets in a path that has m malicious nodes can be represented by $Pr(D_j)$ and can be calculated as:

$$Pr(D_j) = \binom{k}{j} \lambda^j (1 - \lambda)^{k-j}$$

However, the probability of that there is j packets dropped out of k packets sent, with one legitimate packet reaching the destination can be calculated as:

$$\sum_{j=1}^{k-1} Pr(D_j) = \sum_{j=1}^{k-1} \binom{k}{j} \lambda^j (1 - \lambda)^{k-j}$$

So the probability of having a malicious path, which represents the probability that a selective packet dropping attack has happened, given that there is at least one legitimate packet reaching the destination (as assumed in Sect. 3.1) can be written as:

$$Pr(D|\beta) = \frac{\sum_{j=1}^{k-1} \binom{k}{j} \lambda^j (1 - \lambda)^{k-j}}{\beta}$$

The probability of receiving at least one legitimate packet at the destination with 1, 2, and 5 malicious nodes is shown in Figs. 7, 8, and 9 respectively. The mathematical analysis showed that when the number of malicious nodes was low, the probability of receiving at least one legitimate packet at the destination node was high, and when the number of malicious nodes increased, the probability of receiving at least one legitimate packet at the destination node decreased. This was because the higher the number of malicious nodes, the higher the chances of dropping all (k) packets. The results of the probability also showed that the probability of receiving at least one legitimate packet to reach at the destination increased when the transfer of packets increased.

The probability of having a malicious path with 1, 2, and 5 malicious nodes in the path is shown in Figs. 10, 11, and 12. The results of the probability showed that the probability of packet dropping increased when the number of packets increased. The probability of having a malicious path increased with the increasing probability of dropping the packets. With a high number of malicious nodes, the probability of packet dropping would be high, as a result the probability of having a malicious path would be high.

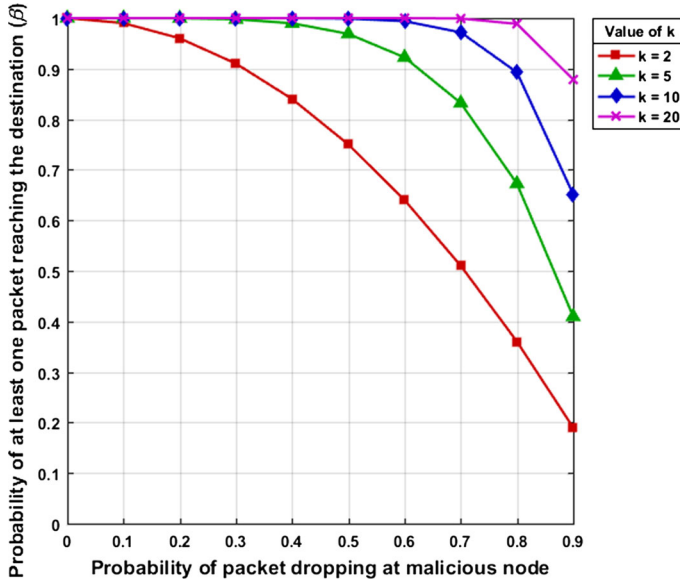


Fig. 7 Probability of receiving at least one legitimate packet at the destination with 1 malicious node in a path

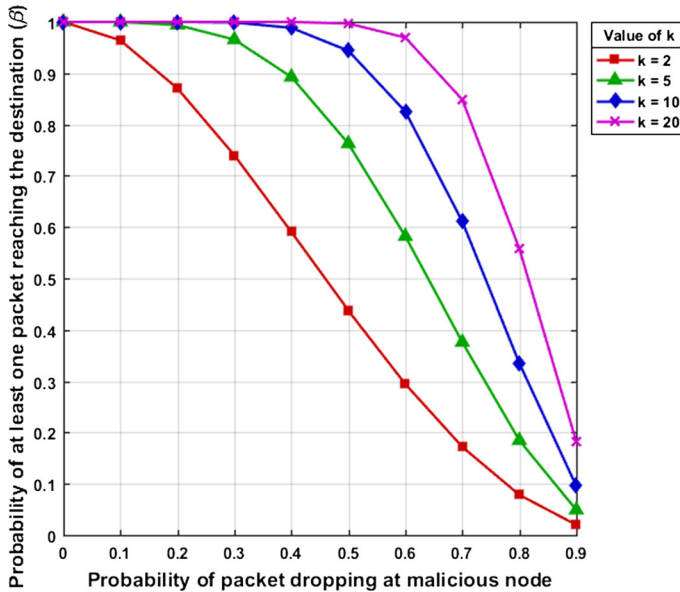


Fig. 8 Probability of receiving at least one legitimate packet at the destination with 2 malicious nodes in a path

The relationship between the number of packets and the probability of at least one legitimate packet being dropped (β) across malicious paths with 1, 2, and 5 malicious nodes in the path is shown in Figs. 13, 14, and 15. In order to detect a malicious node

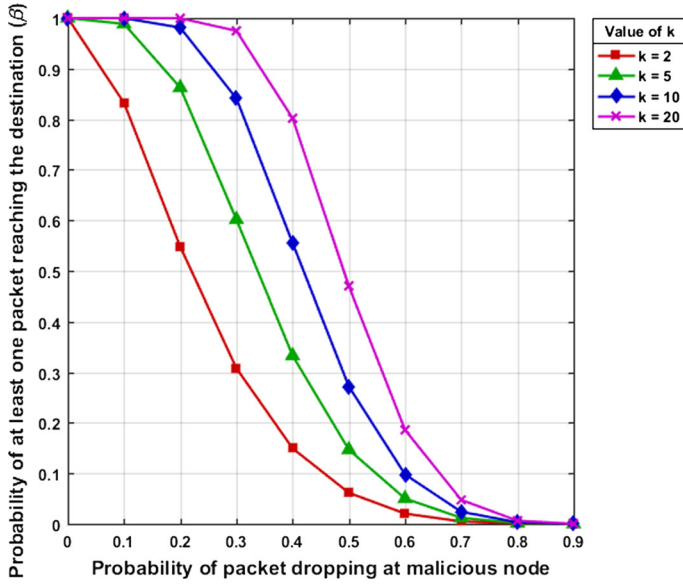


Fig. 9 Probability of receiving at least one legitimate packet at the destination with 5 malicious nodes in a path

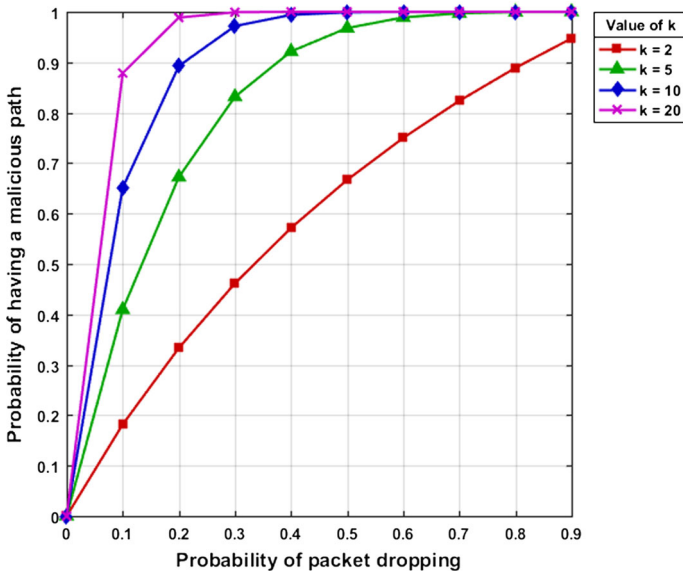


Fig. 10 Probability of detecting a malicious path with 1 malicious node

dropping a packet, the condition of dropping the packet by a malicious node needed to be achieved. A higher probability of packet dropping was achieved when higher number of packets were sent. When increasing the number of malicious nodes, larger number of packets needed to be sent in order to achieve β values of 0.85, 0.90, and 0.95.

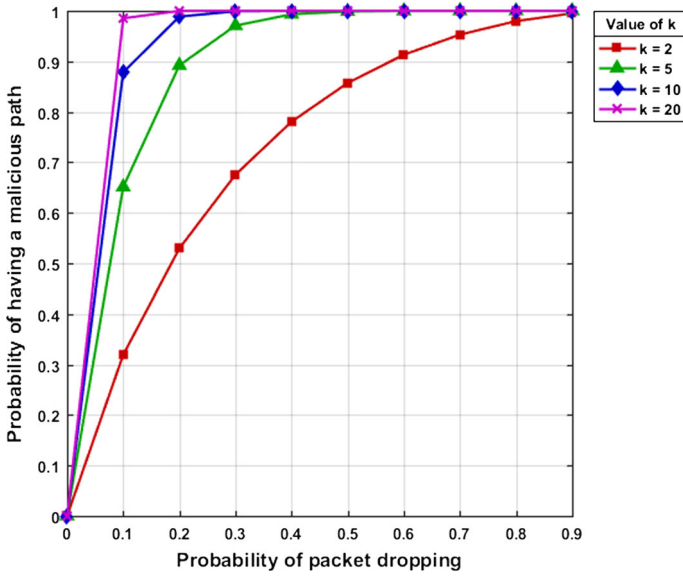


Fig. 11 Probability of detecting a malicious path with 2 malicious nodes

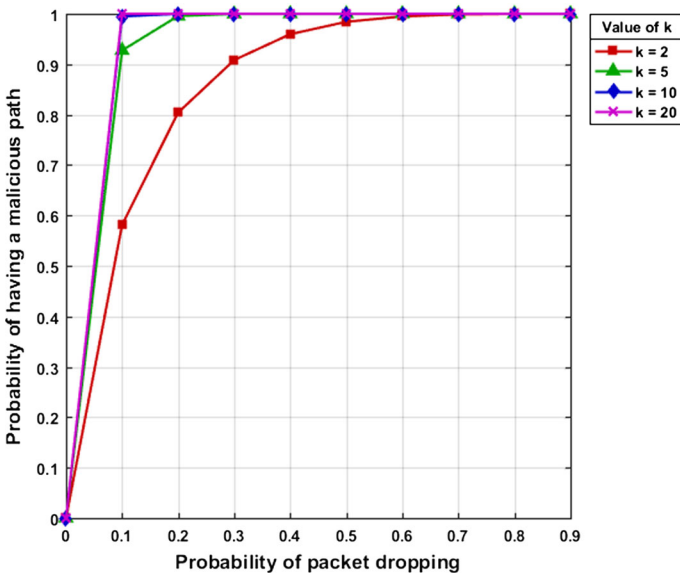


Fig. 12 Probability of detecting a malicious path with 5 malicious nodes

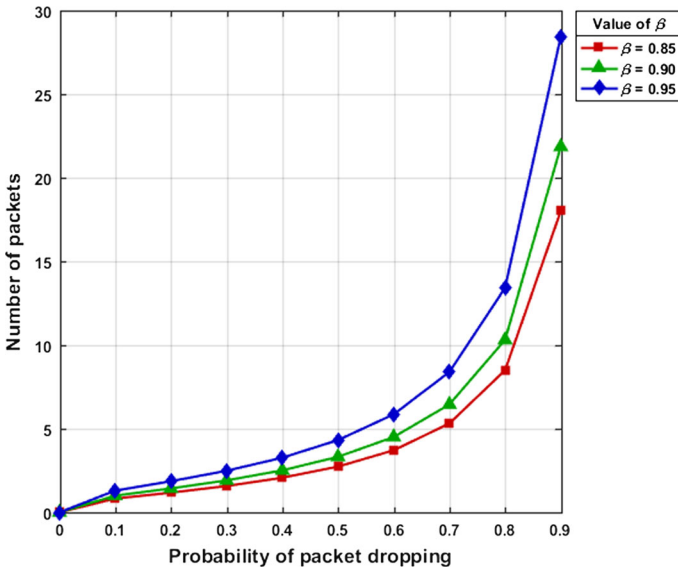


Fig. 13 Number of packets required to accomplish $\beta = (0.85, 0.90, 0.95)$ with 1 malicious node in a path

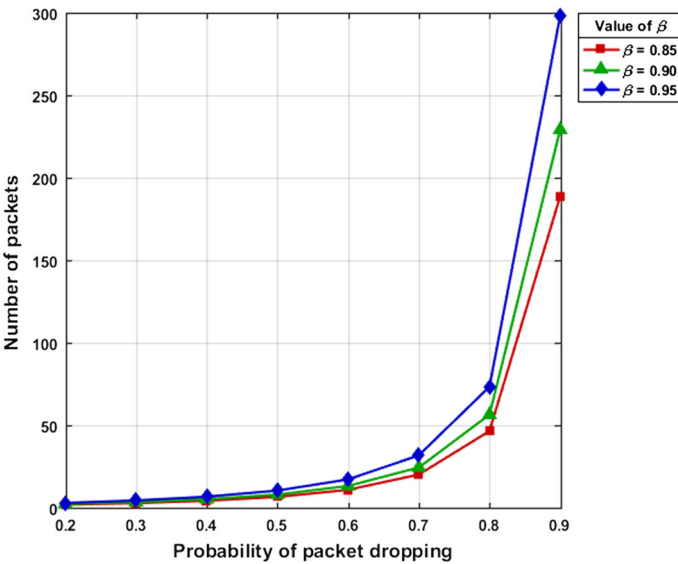


Fig. 14 Number of packets required to accomplish $\beta = (0.85, 0.90, 0.95)$ with 2 malicious nodes in a path

5 Simulation, results and analysis

5.1 Simulation settings

Using the Opportunistic Network Environment simulator (ONE) [28]—designed for OppNets—the developed algorithms were tested for their functionality and effective-

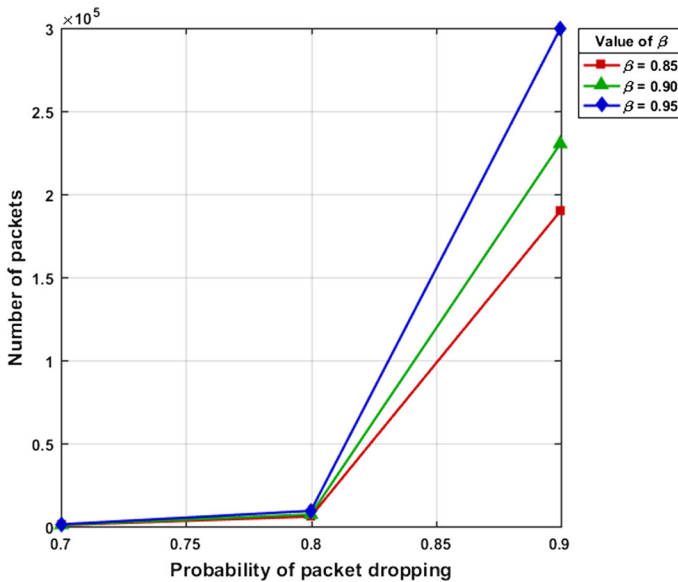


Fig. 15 Number of packets required to accomplish $\beta = (0.85, 0.90, 0.95)$ with 5 malicious nodes in a path

ness. Using the simulator, a scenario was created and run to produce a dataset that was composed of many paths with various nodes with different speeds and types. All the nodes in the generated dataset were legitimate. The dataset was then fed into a written C++ program that manipulated the data by turning legitimate nodes malicious depending on the number of malicious nodes required. The malicious nodes dropped some packets and forwarded the rest to the next hop node. The algorithms were then implemented to detect the malicious nodes.

The settings defined for this scenario follow. The simulation time for this scenario ran for 1 h, 2 h, and then 3 h. Bluetooth was the interface chosen for all nodes transmitted within a range of 10 m at an 8 Mbps speed. The routing protocol used was Epidemic routing.

3 different groups of nodes were defined, each group contained 10 nodes. The first group had only walkers with a walking speed of 1–5 km/h, and buffers with a size of 10MB. The hosts in this group walked on the MapBasedMovement map model. The second group had only cars with driving speeds of 35–60 km/h and 10MB storage space. The hosts in this group drove on roads on the MapBasedMovement map model. The third group contained only trams that drove at a speed of 25–35 km/h, and could store up to 50 MB in their buffers. The hosts in this group drove on the MapRouteMovement map model. Nodes in all groups created new messages with a size ranging between 500 and 700kB every 1–5 s.

5.2 Results and analysis

The following metrics were used to evaluate Algorithms 1, 2, and 3:

1. Malicious path detection accuracy: Defined as the ratio of the total number of detected malicious paths to the total number of actual malicious paths.
2. Malicious node detection accuracy: Defined as the ratio of the total number of correctly detected malicious nodes to the total number of actual malicious nodes.
3. False positive rate: Defined as the ratio of legitimate nodes categorized as malicious nodes.

The malicious path detection accuracy was always 100% for each simulation time because the proposed technique relied on the root values. To achieve a 100% malicious path detection accuracy through evaluation, a path was marked as malicious based on the equality of the Merkle roots. To maintain the 100% path detection accuracy, the destination still ran Algorithms 1, 2, and 3 for nodes with trust values below 0.2. That was done to give an opportunity for good nodes that had been falsely classified as malicious which later appeared in good paths to have their trust values incremented.

The malicious node detection accuracy and the false positive rate for node detection over 1 h, 2 h, and 3 h duration of simulation time are shown in Figs. 16, 17, 18, 19, 20, and 21. The malicious node detection accuracy and the false positive rate for node detection are shown on the y-axis, and the percentage of malicious nodes increasing until all intermediate nodes acted malicious is shown on the x-axis. The results were plotted using the simulation settings defined in Sect. 5.1, but with increasing the simulation time from 1 h to 3 h to show how the results changed over time. Each point on the graph was a result of averaging 30 values resulting from 30 simulation runs.

As can be seen in Figs. 16, 17, and 18, when the simulation time increased, the malicious node had a higher chance to be repeated in paths, so the accuracy was higher. As the number of malicious nodes increased, the algorithms still maintained the accuracy level for detecting the malicious node. Also, the accuracy was its highest when $x = 0.4$, then 0.3 and 0.2, and lowest when $x = 0.1$, but as can be observed next, the increased accuracy came at the cost of increased false positive rate.

It can be observed from Figs. 19, 20, and 21 that when the simulation time increased, the false positive rate also increased. Although the malicious node detection rate increased (in Figs. 16, 17, 18), however, because the algorithms were based on path detection, legitimate nodes were affected as well which resulted in increased false positive rates. With a shorter simulation time, a node had a less chance of being identified as malicious, because the node may have not had the time to appear in enough paths to be falsely classified as malicious. With the increase of malicious node percentage, the false positive rate decreased as the proposed method was relying on path based calculations to build trust among nodes, the nodes became malicious indeed. When $x = 0.1$, the false positive rate was the lowest, and highest when $x = 0.4$. $x = 0.1$ resulted in a constant rate for the 3 h of simulation time, while keeping a good malicious node detection accuracy. When $x = 0.4$, good nodes didn't get the chance to prove themselves legitimate in other paths, increasing the simulation time did not make a difference in this case.

It can be noted from Figs. 16, 17, and 18, that the state of output depended on the value of x . For values where $x = 0.2$ to 0.4, with 2 and 3 h of simulation time, the output was steady because malicious nodes were caught with an increased value of

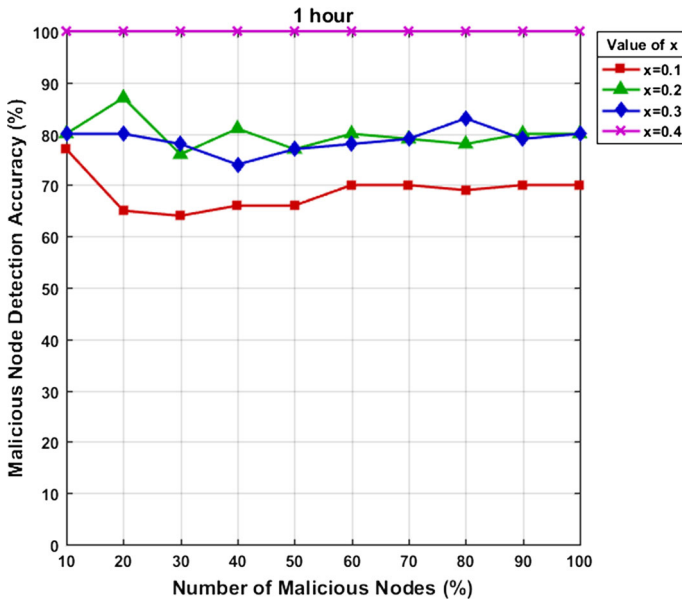


Fig. 16 Malicious node detection accuracy, 1 h of simulation time

x . We found that the detection accuracy increased when the value of x increased and when the simulation time increased, but the increase in node detection accuracy came with an increased false positive rate as noted in Figs. 19, 20, and 21. Based on this, we chose the value of x to be 0.1 when running Algorithm 3 for further tests, because although the accuracy was not the highest (however, very reasonable), but the false positive rate was the lowest.

5.3 Our approach in Epidemic routing vs our approach in PROPHET routing

The malicious node detection accuracy (obtained above in Figs. 16, 17, 18) through Epidemic routing, and obtained and compared with PROPHET routing (with $x = 0.1$ in Algorithm 3) is shown in Figs. 22, 23, and 24. PROPHET achieved an accuracy similar to Epidemic routing, only slightly more. This is because both Epidemic and PROPHET routing replicated multiple copies of the message in the network to increase the chances of message delivery.

It can be noted that with 2 and 3 h of simulation time, the results were almost identical. This is because when $x = 0.1$, nodes that had very slow speed like pedestrians (as defined in Sect. 5.1) didn't get a chance to be repeated in paths even when the simulation time increased, thus when they were malicious, they didn't get detected with a threshold value of 0.2 and $x = 0.1$. However, increasing the value of x gave a full detection accuracy with 2 and 3 h of simulation time (as shown in Figs. 16, 17, 18). However, as explained previously, $x = 0.1$ was chosen because of the low false positive rates.

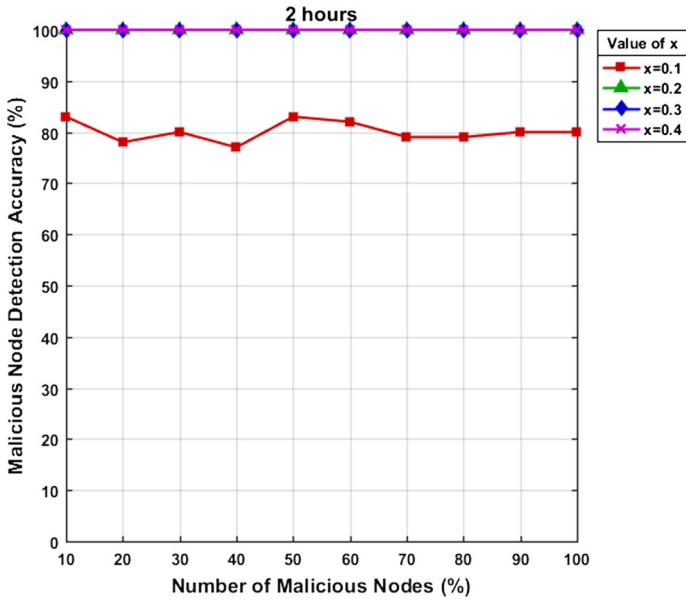


Fig. 17 Malicious node detection accuracy, 2 h of simulation time

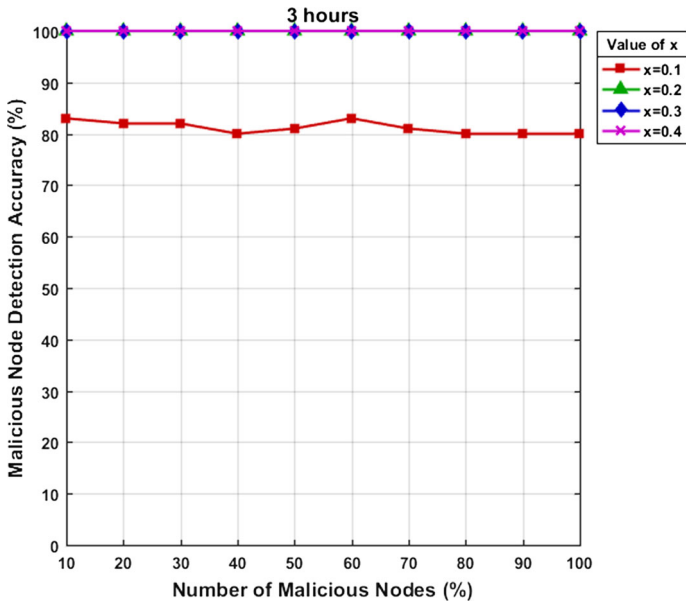


Fig. 18 Malicious node detection accuracy, 3 h of simulation time

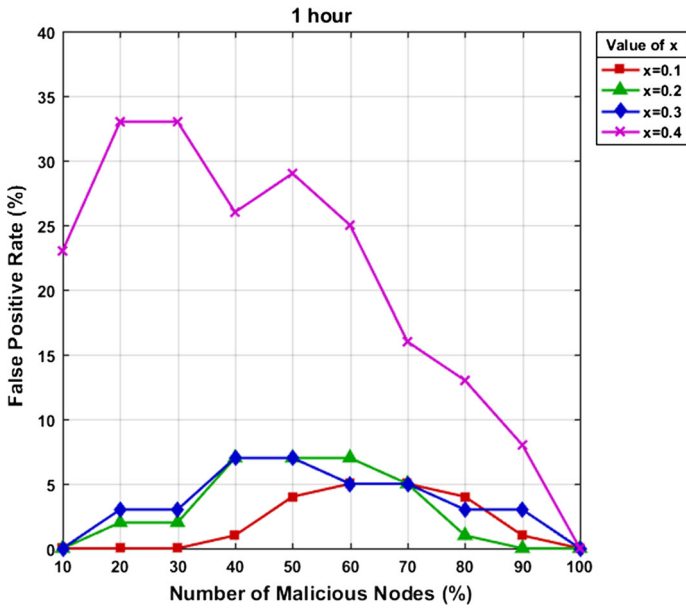


Fig. 19 False positive rate for node detection, 1 h of simulation time

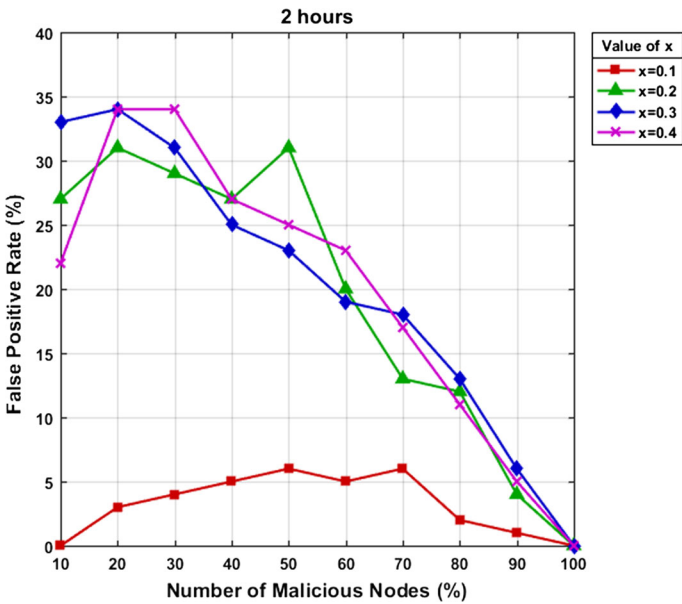


Fig. 20 False positive rate for node detection, 2 h of simulation time

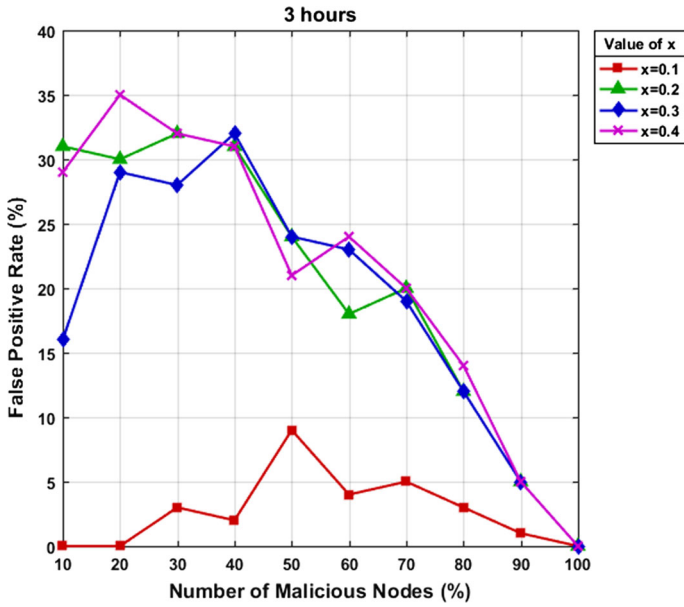


Fig. 21 False positive rate for node detection, 3 h of simulation time

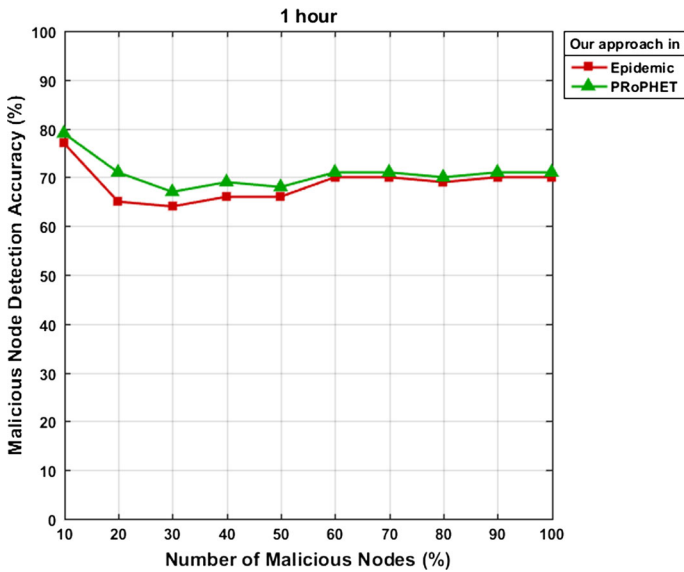


Fig. 22 Malicious node detection accuracy using PRoPHET, 1 h of simulation time

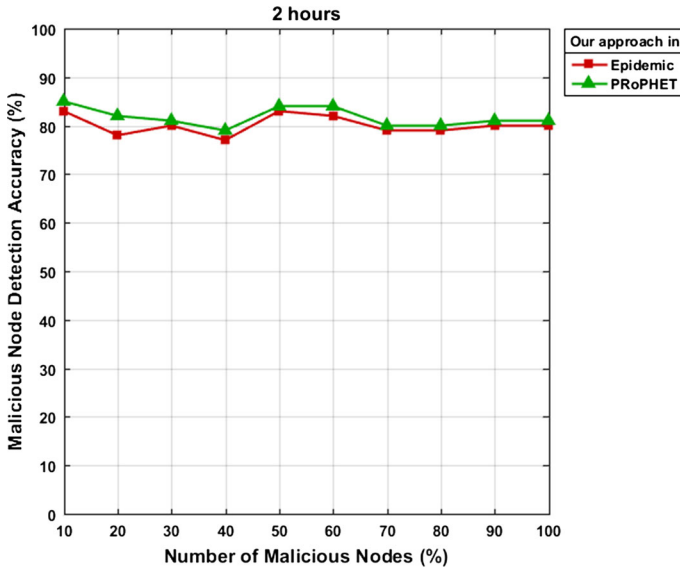


Fig. 23 Malicious node detection accuracy using PRoPHET, 2 h of simulation time

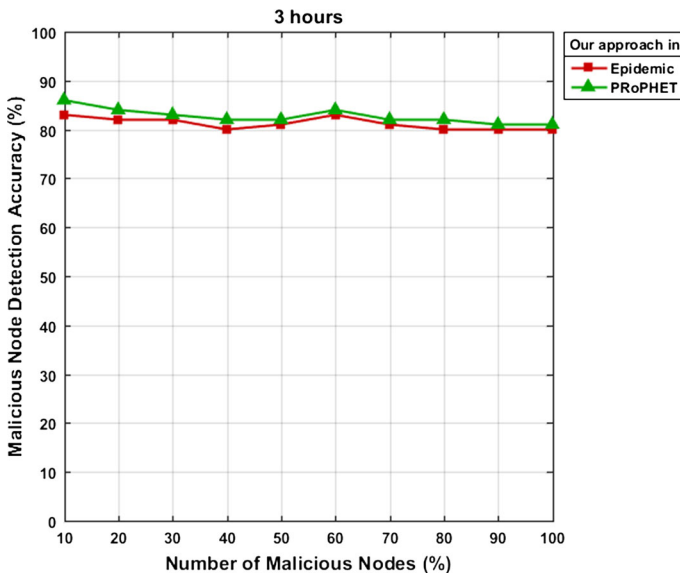


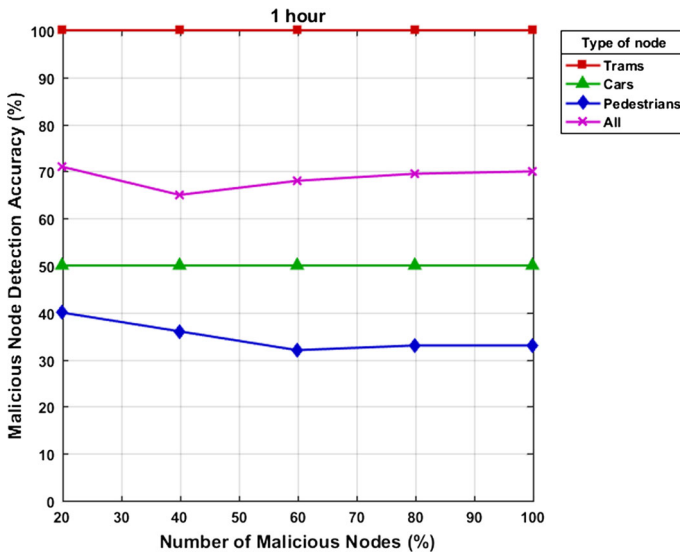
Fig. 24 Malicious node detection accuracy using PRoPHET, 3 h of simulation time

6 Impact of mobility speed and storage on malicious node detection accuracy

The evaluation was extended to study the impact of mobility speed and storage on the malicious node detection accuracy. Algorithms 1, 2, and 3 (with $x = 0.1$ in Algorithm 3)

Table 1 Node speed, transmit speed, transmit range, and buffer settings

Node type	Speed (km/h)	Transmit speed (Mbps)	Transmit range (m)	Buffer space (MB)
Pedestrians	1–5	8	10	10
Cars	35–60	8	10	10
Trams	25–35	8	10	50

**Fig. 25** Malicious node detection accuracy (impact of mobility and storage), 1 h of simulation time

were tested with each type of node (trams, cars, and pedestrians) from 1 h to 3 h of simulation time.

The malicious node detection accuracy, and the false positive rate for node detection are shown in Figs. 25, 26, 27, 28, 29, and 30.

It can be observed from Figs. 25, 26, and 27, that trams achieved a 100% detection accuracy at all times when increasing the percentage of malicious nodes in the network, and when increasing the simulation time. Cars started off with achieving 50% accuracy, and when increasing the simulation time, the accuracy increased to 75% then up to 100%. Pedestrians achieved the lowest detection accuracy that started to decrease with the increase in percentage of malicious nodes. The accuracy did not show much rise over time. Because of their slow speed and limited storage, pedestrians were not repeated in enough paths to be detected as malicious.

It can further be observed from Figs. 28, 29, and 30, that trams always had the highest false positive rates. This is because of their frequent occurrence in paths which resulted in misclassifying other good nodes that existed in the same path. Whereas cars and pedestrians didn't occur in paths as much as trams to mis-classify good nodes as malicious.

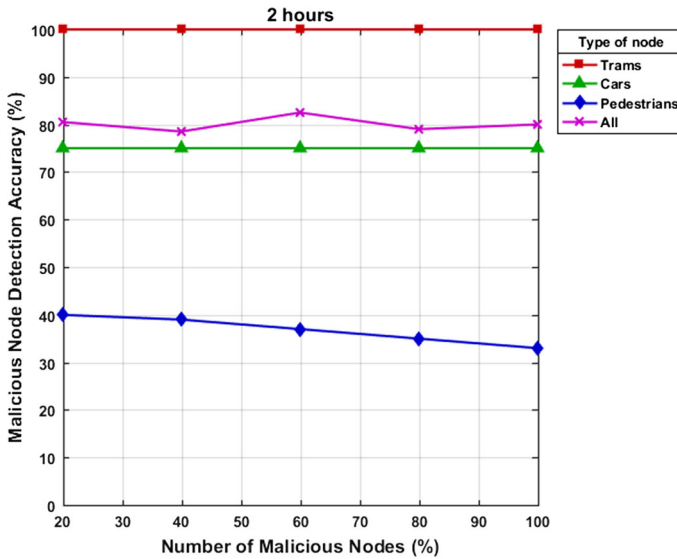


Fig. 26 Malicious node detection accuracy (impact of mobility and storage), 2h of simulation time

The settings used are captured in Table 1, it is noted that although cars drove faster than trams, trams still achieved a higher detection rate because they were able to carry information more than other nodes and this is due to their large buffers. The large buffers gave trams the space for storing messages, thus they accepted more new messages. So trams were chosen by nodes to transfer messages which stayed on their buffers until reaching destination nodes, unlike cars and pedestrians who were more likely to drop the message when their buffers were full. This is why trams reoccurred in multiple paths more than cars and pedestrians did, so when a tram was chosen to be malicious, it meant more paths became malicious.

7 Comparison with other solutions

In this comparison section, a comparison analysis showing the performance of the proposed trust-based approach with other non-trust based OppNet routing approaches is presented. The most popular OppNet protocols are Epidemic routing and PROPHET. These protocols are non-trust based. The comparison was done with non-trust methods for the purpose of showing the level of importance of using trust in an OppNet, and to determine if trust based methods contributed in controlling the packet dropping rate in an OppNet. A quick brief about Epidemic routing and PROPHET follows:

- The basic idea of Epidemic routing (Vahdat and Becker [29]), is that when two nodes meet each other in a network, they first exchange their summary vectors which has a record of all the messages a node is storing. The two nodes then look into the received summary vectors to see which messages they don't have and request those messages from the other node.

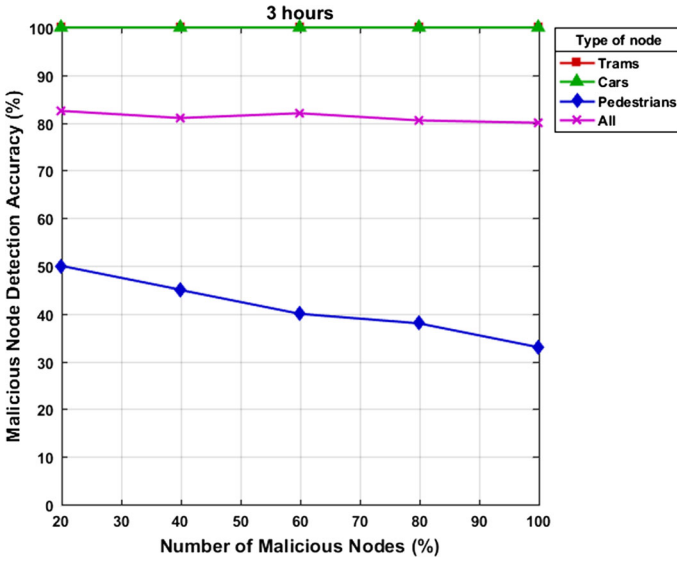


Fig. 27 Malicious node detection accuracy (impact of mobility and storage), 3 h of simulation time

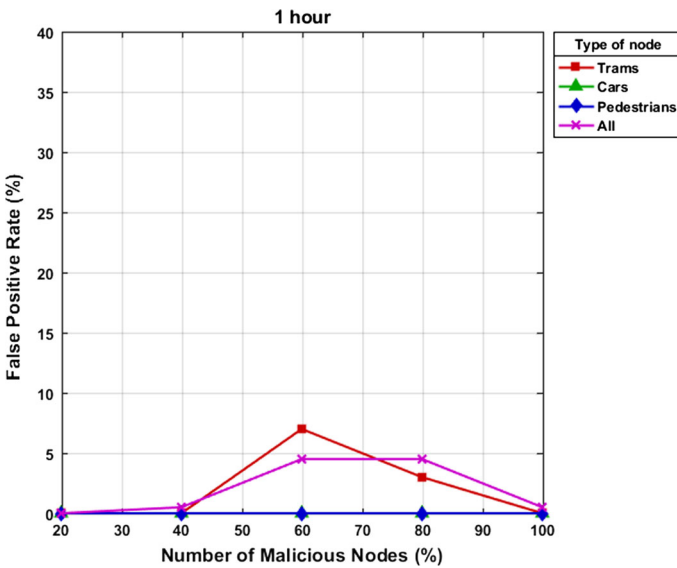


Fig. 28 False positive rate for node detection (impact of mobility and storage), 1 h of simulation time

- PROPHET [30] improves Epidemic routing by adding the delivery predictability (calculated from past records) in the summary vector which gives further information to nodes about the probability of a node to reach the destination. If the delivery predictability of a node is high, then it will get a copy of the message, otherwise it won't.

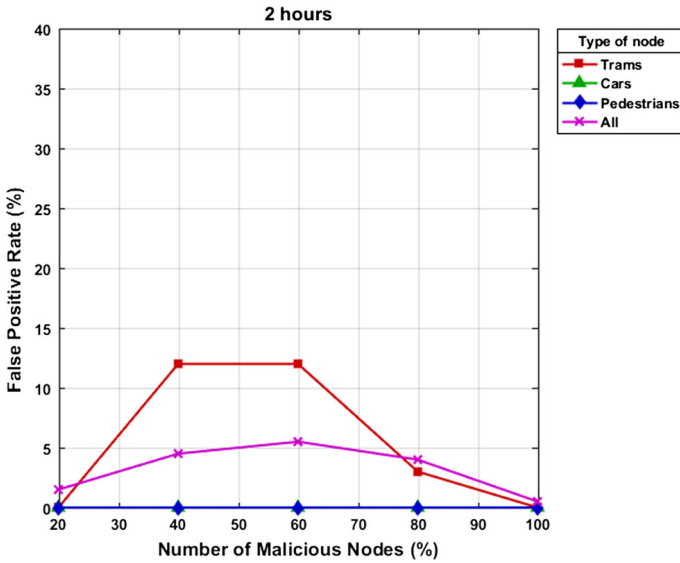


Fig. 29 False positive rate for node detection (impact of mobility and storage), 2h of simulation time

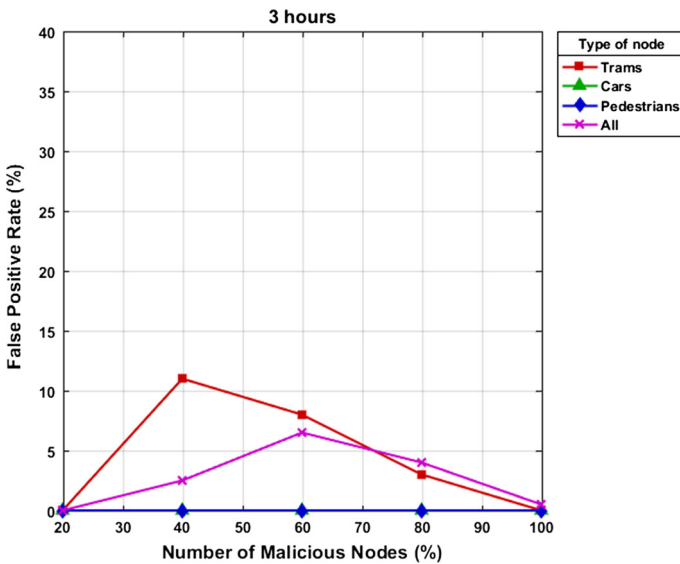


Fig. 30 False positive rate for node detection (impact of mobility and storage), 3h of simulation time

The metric used to compare our technique with the 2 OppNet protocols mentioned above is the packet dropping rate, which is defined as the ratio of the total dropped packets to the total number of packets in the network.

The packet dropping rate over time tested in a network with 50% (50% was chosen because it was the moderate rate) malicious nodes using the developed algorithms (with $x = 0.1$ in Algorithm 3) and 2 non-trust OppNet protocols is shown in Fig. 31.

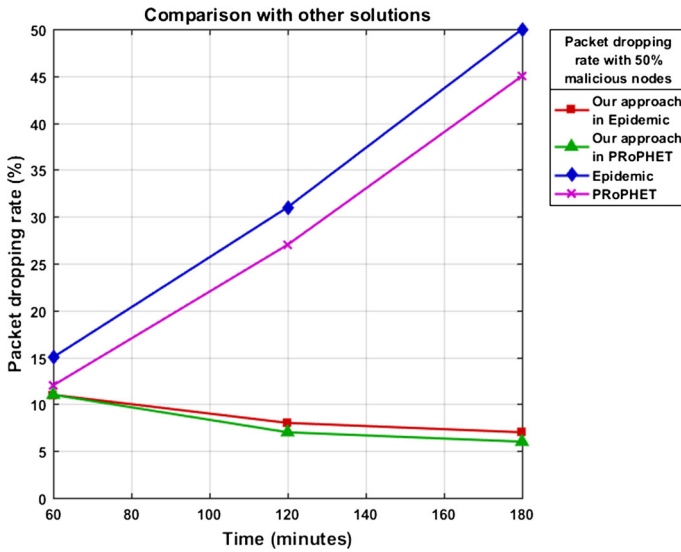


Fig. 31 Comparison between our approach and other non-trust based approaches

It can be noted that under attack conditions with no trust strategy implemented there was no difference in performance between the two routing protocols, and the dropping rate was not controlled. It can be observed that there was significant improvement with our proposed approach implemented with both routing strategies. Hence, the benefits were not inherent or dependent on the routing strategy used. The algorithms worked as they minimized packets dropped when destination nodes identified malicious nodes and prevented them from forwarding incomplete packets. Thus, over time, the dropping rate was controlled and reduced, and this proved the effectiveness of the proposed algorithms. The results show that a trust based model worked more efficiently than a non-trust based model, results have shown how the proposed trust mechanism improved routing by controlling the packet dropping rate.

8 Conclusion and future work

Evolved from DTNs, OppNets have taken a great attention by researches. Research continues to make OppNets better. Securing OppNets is a challenge problem due to their features such as the irregular connections between nodes and the dynamically changing paths. Selective packet dropping attacks develop from malicious intermediate nodes by dropping some packets while transferring them. This is a denial of service attack that may severely degrade the network performance. Malicious path and malicious node detection techniques designed to detect selective packet dropping attacks were presented in this paper. For malicious path and malicious node detection, the techniques used the Merkle tree hashing technique and trust to create trust relationships between nodes and destination nodes. Our simulation results showed that the technique achieved 100% malicious path detection accuracy, and over time, the

technique achieved a high malicious node detection accuracy as intermediate nodes had more time to establish trust with destination nodes. Based on extensive simulation tests, we showed that delivery rates increased with increased storage. Results also showed how our trust mechanism improved routing by controlling the packet dropping rate. We also showed that a trust model worked more efficiently than a non-trust model. Our next step is to adopt the algorithms by intermediate nodes and not only by destination nodes. In addition to direct dynamic trust, reputation will be added to utilize legitimate nodes in the network.

References

1. Tang L, Chai Y, Li Y, Weng B (2012) Buffer management policies in opportunistic networks. *J Comput Inf Syst* 8(12):5149–5159
2. Chaintreau A, Hui P, Crowcroft J, Diot C, Gass R, Scott J (2005) Pocket switched networks: real-world mobility and its consequences for opportunistic forwarding. Technical Report, UCAM-CL-TR-617, University of Cambridge, Feb 2005
3. Juang P, Oki H, Wang Y, Martonosi M, Peh L, Rubenstein D (2002) Energy-efficient computing for wildlife tracking: design tradeoffs and early experiences with ZebraNet. *Proc Tenth Int Conf Archit Support Progr Lang Oper Syst* 37(10):96–107
4. Rigano C, Scott K, Bush J, Edell R, Parikh S, Wade R, Adamson B (2008) Mitigating naval network instabilities with disruption tolerant networking. In: IEEE conference on military communications, Nov 2008, pp 1–7
5. Burleigh S, Hooke A, Torgerson L, Fall K, Cerf V, Durst B, Scott K, Weiss H (2003) Delay-tolerant networking: an approach to interplanetary internet. *IEEE Commun Mag* 41(6):128–136
6. Ahmad A, Alajeely M, Doss R (2014) Defense against packet dropping attacks in opportunistic networks. In: Advances in computing, communications and informatics (ICACCI, 2014 IEEE international conference), Sept 2014, pp 1608–1613
7. Lee S, Gerla M (2001) Split multipath routing with maximally disjoint paths in ad hoc networks. *IEEE Int Conf Commun* 10:3201–3205
8. Baadache A, Belmehdi A (2012) Fighting against packet dropping misbehaviour in multi-hop wireless ad hoc networks. *J Netw Comput Appl* 35(3):1130–1139
9. Zhang X, Jain A, Perrig A (2008) Packet-dropping adversary identification for data plane security. In: Proceedings of the 2008 ACM CoNEXT Conference, Dec 2008
10. Chuah M, Yang P, Han J (2007) A Ferry-based intrusion detection scheme for sparsely connected ad hoc networks. In: Proceedings of the 2007 fourth annual international conference on mobile and ubiquitous systems: networking services, Aug 2007, pp 1–8
11. Shang-Fu G, Jian-Lei Z (2012) A survey of reputation and trust mechanism in peer-to-peer network. In: Proceedings of the 2012 international conference on industrial control and electronics engineering, Aug 2012, pp 116–119
12. Pirezada A, Datta A, McDonald C (2004) Trusted routing in ad-hoc networks using pheromone trails. *IEEE Congr Evol Comput* 2:1938–1943
13. Yajun G, Yulin W (2007) Establishing trust relationship in mobile ad-hoc network. In: International conference on wireless communications, networking and mobile computing, Sept 2007, pp 1562–1564
14. Yu Y, Li K, Zhou W, Li P (2012) Trust mechanisms in wireless sensor networks: attack analysis and countermeasures. *J Netw Comput Appl* 35(3):867–880
15. Chen I, Bao F, Chang M, Cho J (2014) Dynamic trust management for delay tolerant networks and its application to secure routing. *IEEE Trans Parallel Distrib Syst* 25(5):1200–1210
16. Gonçalves M, Moreira E, Martimiano L (2010) Trust management in opportunistic networks. In: Proceedings of the 2010 ninth international conference on networks, April 2010, pp 209–214
17. Merkle R (1980) Protocols for public key cryptosystems. *IEEE Symp secur priv* 1109:122–134
18. Li Y, Gouda M (2008) Sources and monitors: a trust model for peer-to-peer networks. In: Proceedings of seventeenth international conference on computer communications and networks, Aug 2008, pp 1–6

19. Cho J, Chen I (2016) PROVEST: provenance-based trust model for delay tolerant networks. *IEEE Trans Dependable Secur Comput* 99:1–1
20. Nguyễn C, Camp O (2008) Using context information to improve computation of trust in ad hoc networks. In: *Proceedings of the 2008 IEEE international conference on wireless and mobile computing, networking and communications*, Oct 2008, pp 619–624
21. Bijon K, Haque M, Hasan R (2014) A trust based Information sharing model (TRUISM) in MANET in the presence of uncertainty. In: *Twelfth annual international conference on privacy, security and trust (PST)*, July 2014, pp 347–354
22. Yao L, Man Y, Huang Z, Deng J, Wang X (2016) Secure routing based on social similarity in opportunistic networks. *IEEE Trans Wirel Commun* 15(1):594–605
23. Zia T, Islam M (2010) Communal reputation and individual trust (CRIT) in wireless sensor networks. In: *Proceedings of the 2010 international conference on availability, reliability and security*, Feb 2010, pp 347–352
24. Marti S, Giuli T, Lai K, Baker M (2000) Mitigating routing misbehaviour in mobile ad hoc networks. In: *Proceedings of the sixth annual international conference on mobile computing and networking*, pp 255–265
25. Jaimes L, Ullah K, Moreira E (2016) ARS: anonymous reputation system for vehicular ad hoc networks. In: *Eighth IEEE Latin-American conference on communications (LATINCOM)*, Nov 2016, pp 1–6
26. Dini G, Duca A (2012) Towards a reputation-based routing protocol to contrast blackholes in a delay tolerant network. *Ad Hoc Netw* 10(7):1167–1178
27. DeCandia G, Hastorun D, Jampani M, Kakulapati G, Lakshman A, Pilchin A, Sivasubramania S, Voshall P, Vogels W (2007) Dynamo: amazon’s highly available key-value store. In: *Proceedings of twenty first ACM SIGOPS symposium on operating systems principles*, Oct 2007, pp 205–220
28. ONE version 1.4.1. <http://www.netlab.tkk.fi/tutkimus/dtn/theone/>
29. Vahdat A, Becker D (2000) Epidemic routing for partially-connected ad hoc networks. Technical Report, CS-200006, Duke University
30. Lindgren A, Doria A, Schelén O (2003) Probabilistic routing in intermittently connected networks. *ACM SIGMOBILE Mob Comput Commun Rev* 7(3):19–20