Analyzing, Quantifying, and Detecting the Blackhole attack in Infrastructure-less Networks

Christoforos Panos[1], Chirstoforos Ntantogian[2], Stefanos Malliaros[2], Christos Xenakis[2]

[1]Department of Informatics & Telecommunications, University of Athens, Greece

cpanos@di.uoa.gr

[2]Department of Digital Systems, University of Piraeus, Greece

{dadoyan, stefmal, xenakis}@unipi.gr

Abstract

The blackhole attack is one of the simplest yet effective attacks that target the AODV protocol. Blackhole attackers exploit AODV parameters in order to win route requests, and thus, attract traffic, which they subsequently capture and drop. However, the first part of the attack is often neglected in present literature, while the majority of attempts in detection focus only on the second part of the attack (i.e., packet drop). This paper provides a comprehensive analysis of the blackhole attack, focusing not only on the effects of the attack, but also on the exploitation of the route discovery process. As a result, a new critical attack parameter is identified (i.e., blackhole intensity), which quantifies the relation between AODV's sequence number parameter and the performance of blackhole attacks. In addition, a novel blackhole detection mechanism is also proposed. This mechanism utilizes a dynamic threshold cumulative sum (CUSUM) test in order to detect abrupt changes in the normal behavior of AODV's sequence number parameter. A key advantage of the proposed mechanism is its ability to accurately detect blackhole attacks with a minimal rate of false positives, even if the malicious node selectively drops packets.

Keywords: Blackhole attack, MANET, AODV, IDS, CUSUM.

1 Introduction

Infrastructure-less networks have gained considerable popularity due to the recent proliferation of mobile computing (i.e., smart phones, tablets, etc.). These networks comprise a wide range of networking paradigms such as mesh networks, mobile ad hoc networks (MANETs), vehicular ad hoc networks (VANETs), delay tolerant networks (DTN), opportunistic and sensor networks [1]. A common characteristic of these networks is the absence of any fixed architectural components such as routers, access points, etc., supporting and serving dynamic topologies and behaviors. To accommodate this characteristic, a variety of routing protocols have been proposed in the literature, however, the most widely adopted protocol is the ad hoc on demand distance vector (AODV) routing protocol [2]. AODV is a dynamic and adaptive routing protocol that has been initially designed for MANETs and later adopted in DTN [3], opportunistic [4], mesh [5], as well as sensor networks [6].

AODV operates with the assumption that all participating nodes are well-behaved, and thus, it does not include any security mechanism. Considering also the deployment characteristics of infrastructure-less networks (i.e., wireless shared access, dynamic topologies, cooperative routing, etc.), it can be realized that AODV faces a wide set of security threats [7]. One of the simplest yet effective attack that targets the AODV protocol is the blackhole attack [7]. The goal of this attack is for a malicious node to attract network traffic through an exploit in the route discovery process, and, subsequently, drop any data packets that are forwarded to it. To this end, the attack is carried out in two steps: in the first step, during a route discovery process, the malicious node will modify a critical AODV protocol field in order to falsely advertise itself as the most up to date path to the destination (and thus win all of the received route requests); while in the second step, during the transmission of data by legitimate nodes, it will drop any data packets that are forwarded to it.

To address blackhole attacks, several detection mechanisms have been proposed in the literature [8] (i.e., analyzed in sect. 2.3 of this paper). However, the majority of these mechanisms attempt to resolve if a blackhole attack takes place based on the second step of the attack (i.e., packet drop) and thus, they are effective only when the malicious node indiscriminately drops all the forwarded traffic. Attacks are not detected when they are initiated, but only when a malicious node begins dropping packets (i.e., non-real-time detection). Furthermore, in order to assess if packets are forwarded or dropped, monitoring nodes have to exchange audit data and are thus hindered by communication and storage overheads, as well as increased energy consumption. Finally, in cases of high nodes' mobility or continuous changes in network topology, the collected audit data might lead to inconclusive or erroneous assessments, resulting in false positives/negatives. Based on these observations, we need to identify new parameters that can be used for detection during the first step of the attack (i.e., route discovery process), which is often neglected in the literature. A detection mechanism that relies on such parameters will then be capable of detecting a blackhole attack immediately, even if the malicious node selectively drops data packets. Furthermore, this mechanism will alleviate the need for collecting audit data that may be malicious, incomplete, or outdated, providing an accurate and real time view of the route discovery process. Thus, blackhole attacks will be detected with low false positives/negatives and without the associated overheads of audit data collection.

In this paper, we identify a new critical parameter associated with the first step of a blackhole attack, named as blackhole intensity, and, subsequently, propose a novel blackhole detection mechanism. The blackhole intensity parameter quantifies the relation between AODV's sequence numbers (SQN) and blackhole attacks. Through an extensive set of simulations, we utilize this parameter in order to ascertain: (i) the impact of SQN exploitation on the performance of blackhole attacks (i.e., the percentage of routes won during the route

request process and the percentage of packets dropped, by a malicious node); and (ii) the optimal blackhole intensity parameter values that can be used by a malicious node, in order to maximize the outcome of an attack. Furthermore, the quantitative relation between SQNs and blackhole attacks, exemplified by the blackhole intensity parameter, leads to the conclusion that detection can be achieved by monitoring the behavior of SQNs. Based on this analysis, we propose and evaluate a highly accurate detection mechanism that is capable of detecting blackhole attacks during the first step of the attack. This mechanism is implemented at each network node and utilizes a dynamic threshold cumulative sum (CUSUM) test to detect abrupt changes in the normal behavior of SQNs. As a result, it can accurately detect blackhole attacks with minimal delay, even if the malicious node selectively drops packets. The performance of the proposed detection mechanism is also evaluated through an extensive set of simulations. The numerical results show that it resolves attacks with high detection accuracy and a minimal rate of false positives, even under conditions of high network volatility, while alleviating the need for communication overheads, often exhibited by other detection mechanisms that have been proposed in the literature. Overall the contributions of this paper are threefold:

- We define, categorize and comprehensively analyze blackhole attacks through numerical examples. Moreover, we provide new insights for the blackhole attack by proposing and evaluating a parameter named blackhole intensity that quantifies the impact of SQNs on the blackhole attack.
- We propose and analyze a detection mechanism for blackhole attacks based on CUSUM that does not impose significant overheads or extensive modifications to the AODV routing protocol.
- We implement and comprehensively evaluate our detection mechanism through an extensive set of simulations. We compare our mechanism with previous solutions and show that it outperforms them in terms of detection accuracy, computational cost, and communication overhead.

The rest of this paper is organized as follows. Section 2 analyzes the functionality of the AODV routing protocol and outlines the implementation aspects of the blackhole attack. In section 3, we first analyze the operation of the blackhole attack, in order to outline the critical AODV protocol parameters that are exploited by a malicious node. Then, we define the blackhole intensity parameter, and, subsequently, perform an extensive set of simulations, in order to assess the impact that the blackhole intensity parameter has in the attack, under a variety of different network conditions (i.e., node mobility, variable packet traffic, etc.). Furthermore, we identify the optimal blackhole intensity parameter values that can be used by a malicious node. In section 4, the proposed detection mechanism is presented and

comparatively evaluated with other security mechanisms through simulations. Finally, section 5 contains the conclusions.

2 Background

In this section, we first provide an overview of the AODV protocol's functionality. This overview covers only the most critical aspects of the protocol's operations, since a more detailed analysis of AODV exists in [2]. In section 2.2, we outline the operation of the blackhole attack and all of its associated steps, while in section 2.3; we provide an evaluation of several security solutions for blackhole attacks that have been previously proposed. A comprehensive analysis of all the related literature requires an extensive review, which is outside the scope of this paper.

2.1 Overview of the AODV routing protocol

AODV is an on demand routing protocol, which maintains routes as long they are needed by source nodes. It is scalable and offers low processing, memory, and communication overheads to the underlying network. It utilizes three control messages to achieve route discovery: route request (RREQ), route reply (RREP), and route error (RERR). It also provides an optional fourth control message (i.e., Hello message), which is used for preserving connectivity between neighboring nodes. Each node maintains a list of previously established routing paths in a routing table. Each entry in this table stores routing information to a destination node in the network. The most essential fields of a routing table entry are:

- Destination IP address (dst): the IP address of the destination node.
- Destination SQN (denoted as $SQN_{dst\_node\_entry}$): this is the latest SQN of the destination node of the entry. This field can be updated during the route discovery process. The destination SQN is a measure of the freshness of the routing information in the related entry.
- Hop count (hop_count): represents the current distance to the destination node of the entry.
- Next hop node (next_hop): all packets sent to the destination node of the entry should be forwarded through this node.

When a source node S wishes to transmit a data packet to some destination D for which it does not possess a route, it initiates a route discovery process by first incrementing its own SQN by one, and, subsequently, broadcasting a RREQ message that includes the: source IP address, source SQN, destination IP address, destination SQN, RREQ id, and hop count field. The value of the destination SQN in the RREQ message (the values of destination SQNs in the AODV messages are denoted as SQN_{dst_node}) is taken from the related routing table entry of the source node for the specific destination that wishes to discover a route. The intermediate node

that receive the RREQ first create a routing table entry for the source node S. Then, it checks the routing table for a route to the destination node D. If it possesses a fresh route to the destination (i.e., the $SQN_{dst\_node\_entry}$ in its corresponding routing table entry is greater than or equal to the SQN_{dst_node} included in the RREQ message), then it responds to the source node with a route reply (RREP) that includes: the hop count to the destination, the destination IP address, the destination SQN, and the source IP address (i.e., the address of the node that initiated the route request). The value of the destination SQN (i.e., SQN_{dst_node}) is taken from the stored in the intermediate nodes' routing table. Otherwise, (i.e., if the $SQN_{dst\_node\_entry}$ in the intermediate nodes' routing table entry is less than the SQN_{dst_node} included in the RREQ message or there is no route to the destination at all), then the intermediate node increments the hop count field by one and forwards the RREQ to its neighbors.

If none of the intermediate nodes possesses a fresh route to the destination, then the RREQ eventually reaches the destination node. In this case, the destination node increases its own SQN by one (if the incremented value equals the value in the RREQ message) and then sends a RREP message to the source node S that contains the: source IP address, destination IP address, destination SQN, and hop count field. The destination SQN (i.e., SQN_{dst_node}) in the RREP message is equal to the value of the destination node's own SQN. Intermediate nodes receiving the RREP update their routing tables, only, if the destination SQN_{dst_node} in the message is higher from the stored value in their routing tables (i.e., $SQN_{dst\_node\_entry}$), or the destination SQNs are equal, but the hop count field in the RREP is smaller than the stored value. If multiple RREP messages reach the source node (i.e., this may occur when several intermediate nodes have a routing path to the destination node), it accepts the RREP with the highest destination SQN value or, in case these values are equal, the RREP with the smallest number of hops to the destination. If a link breaks, an intermediate node initiates a local repair mechanism attempting to discover a new route to the destination, by transmitting a RREQ message. If the repair mechanism fails to discover a route, the node generates a RERR message that includes the IP addresses and the last known destination SQNs of the unreachable destinations, informing the receiving nodes that they should restart the routing discovery process, if they want to communicate with them.

2.2 The blackhole attack

The blackhole attack is a type of denial-of-service attack in which a malicious node falsely claims to possess a fresh route to the destination, in order to attract network traffic, and, subsequently, drops all data packets that are forwarded to it. In a more advanced variation of the attack, the malicious node may even selectively drop a percentage of packets (instead of all), in order to avoid detection. This variation is often referred as greyhole attack [25]. The implementation of the attack can be achieved in two ways, which we refer as "reactive" and

"proactive". In the "reactive" version of the attack, a malicious node awaits for RREQ messages. When it receives an RREQ, then it responds to the source node with a spurious RREP message that includes a fake destination SQN (i.e., $SQN_{malicious}$) of an arbitrarily high value. Upon receiving the fake RREP message, the source node compares the $SQN_{malicious}$ value with the SQN values of any other received RREP messages, and, since $SQN_{malicious}$ has the highest value; the source node selects the malicious node as its path to the destination. Subsequently, the source node begins the transmission of data through the malicious node.

In the "proactive" version of the attack, a malicious node actively generates fake RREQ messages, masquerading as an intermediate node forwarding a RREQ message. First, it selects a random source and destination address and then, it generates and transmits a RREQ message that includes a fake source SQN of arbitrarily high value. Upon receiving the fake RREQ message, intermediate nodes add the malicious node as a path to the destination. Subsequently, when they have data to transmit to the destination, they select the malicious node as a path to the destination. The "proactive" version of the attack can yield more captured traffic for the malicious node, since: (i) the later does not have to wait for RREQ messages in order to advertise its spurious path to the destination; and (ii) it enables the malicious node to actively advertise a path to any destination, contrary to the "reactive" version of the attack, where the malicious node is limited to the destinations from which a RREQ message is received.

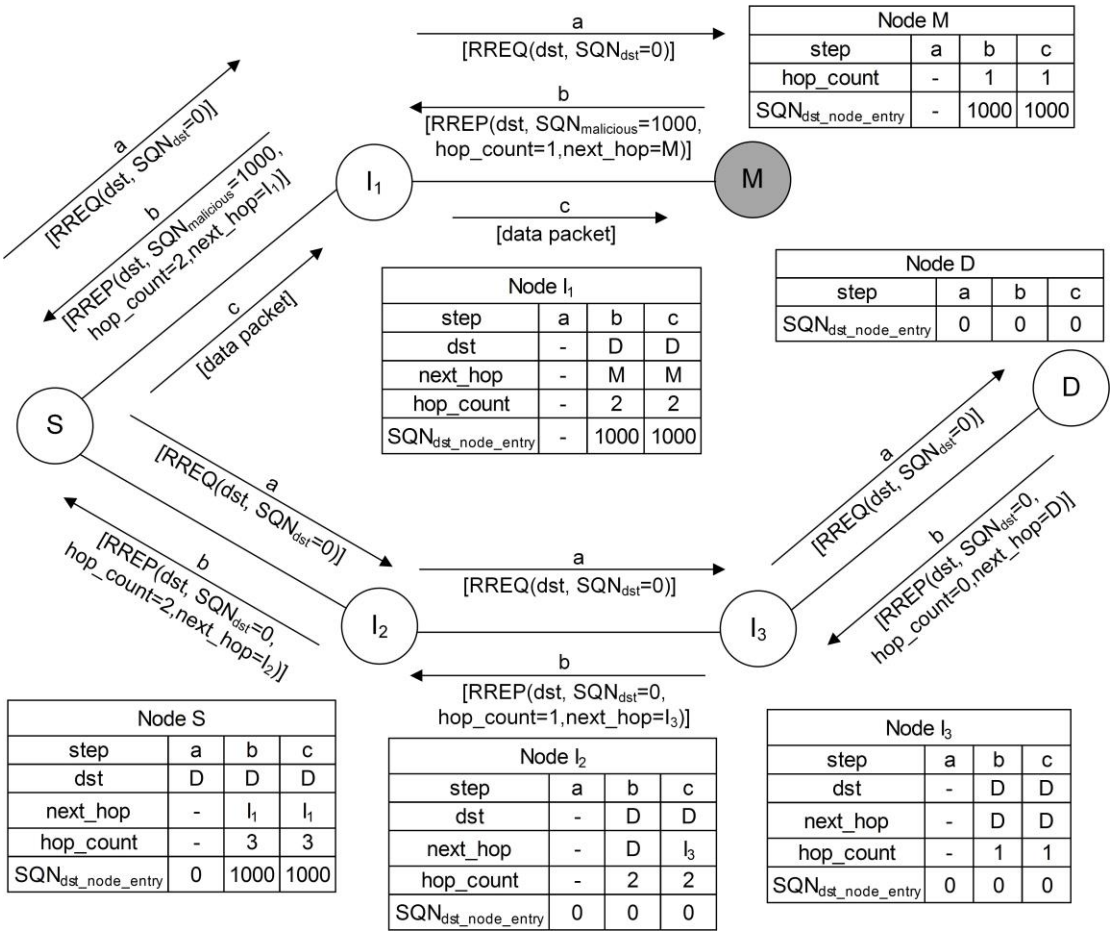
On the other hand, detecting the "proactive" version of the attack can be implemented using a simple mechanism that takes advantage of the AODV operation. This detection mechanism should run in every node and simply check if a received RREQ message was actually generated and transmitted by the host node itself. In particular, according to the AODV protocol specifications, when a node on the network receives a RREQ message, it compares the source IP address and RREQ id with any values stored in its buffer, in order to avoid processing RREQ messages that have already been processed or that have been transmitted by itself [2]. If no matching values are found (i.e., the RREQ message is new to the host node) then the detection mechanism checks if the source IP address on the RREQ message matches the IP address of the host node. If the two IP address values match, then the RREQ message has been generated by a malicious node (even though the host node is listed as the source in the RREQ message's header) and, thus, a “proactive” blackhole attack has been detected. Consequently, as we have shown, the “proactive” version of the attack can be detected by intelligently performing only one additional comparison by the detection mechanism, thus inducing insignificant computational overhead to the host node. For this reason, throughout the remainder of this paper, we focus on the "reactive" version of the blackhole attack.

To better understand the functionality of a "reactive" blackhole attack, we provide a numerical example that presents all of the steps taken by a malicious node. Figure 1 shows a network of six nodes. Node S denotes the source node, node D the destination node, nodes I_1,

I_2, I_3 are intermediate nodes; while node M is the malicious node performing a blackhole attack. When node S wants to transmit data to the destination, it first checks for a valid route to its routing table. Since no such route exists, node S generates a RREQ message (with parameters dst = D, $SQN_{dst\_node} = 0$) and transmits it to its neighboring nodes I_1 and I_2, (see Figure 1, step a). These nodes do not possess a route to the destination yet either, so the RREQ message is subsequently forwarded, and, finally, it reaches both the malicious node M and the destination node D.

Upon the reception of the RREQ message, the malicious node M, generates a RREP message (even though it does not possess a route to the destination node D), using as a destination SQN_{dst_node} (which is denoted as $SQN_{malicious}$) an arbitrarily high value, 1000 in our example, as well as a fake hop_count = 1, and transmits the message to the next hop (i.e., node I_1) towards the source node S (see Figure 1, step b). The intermediate node I_1 that receives the RREP message generated by the malicious node M; creates a new route table entry for the destination, in which it stores the destination address (dst), next_hop, hop_count incremented by one, and the fake $SQN_{malicious}$ value from the RREP message to its $SQN_{dst\_node\_entry}$ field. Subsequently, it updates the received RREP message with the incremented hop_count and with the next_hop field set equal to its own address. Finally, it forwards the RREP message towards the source node S. When the source node S receives the RREP message, it creates a new route table entry for the destination, in which it stores the destination address (dst), next_hop, hop_count incremented by one, and the fake $SQN_{malicious}$ value from the RREP message to the $SQN_{dst\_node\_entry}$ field.

On the other hand, the destination node D generates a RREP message (with parameters $SQN_{dst\_node} = 0$, hop_count = 0) and transmits it to the next hop (i.e., node I_3) towards the source node S (see Figure 1, step b). Each of the intermediate nodes (i.e., I_3 and I_2) that receive the RREP message generated by the destination node D, create a new route table entry for the destination, in which they store the destination address (dst), next_hop, hop_count incremented by one, and SQN_{dst_node} value from the RREP message. Subsequently, they update the received RREP message with the incremented hop_count and with the next_hop field set equal to their own address. Finally, they forward the RREP message towards the source node S. When the source node S receives the RREP message generated by the destination node D, it compares the SQN value between the entry stored in the route table (i.e., $SQN_{dst\_node\_entry}$) and the value in the RREP message (i.e., SQN_{dst_node}) and, since the later contains a lower value, the RREP message is discarded.



Node M

step	a	b	c
hop_count	-	1	1
SQN$_{dst_node_entry}$	-	1000	1000

Node I$_1$

step	a	b	c
dst	-	D	D
next_hop	-	M	M
hop_count	-	2	2
SQN$_{dst_node_entry}$	-	1000	1000

Node D

step	a	b	c
SQN$_{dst_node_entry}$	0	0	0

Node S

step	a	b	c
dst	D	D	D
next_hop	-	I$_1$	I$_1$
hop_count	-	3	3
SQN$_{dst_node_entry}$	0	1000	1000

Node I$_2$

step	a	b	c
dst	-	D	D
next_hop	-	D	I$_3$
hop_count	-	2	2
SQN$_{dst_node_entry}$	0	0	0

Node I$_3$

step	a	b	c
dst	-	D	D
next_hop	-	D	D
hop_count	-	1	1
SQN$_{dst_node_entry}$	0	0	0

Figure 1: the "reactive" blackhole attack (step a: route request, step b: route replies, step c: data transmission)

Once the route discovery process is completed, the source node S looks up its route table for the next_hop node of destination D (i.e., node I_1) and transmits a data packet to it (see Figure 1, step c). Subsequently, node I_1 receives the data packet and checks if the packet is addressed for itself. Since the data packet destination field indicates that the message's destination is node D, node I_1 looks up its route table for the next_hop node of destination D (i.e., node M) and forwards the data packet to it. Finally, once the malicious node M receives the data packet, it can perform one of the two possible actions: it either (i) arbitrarily drops the data packet, or (ii) selectively drops the packet based on a percentage of target packet drops.

2.3 Related work

The blackhole attack has been repeatedly analyzed in the literature. In [9], the authors provide an overview of routing attacks that target MANETs, including the blackhole attack. Furthermore, the authors survey several detection mechanisms that attempt to address blackhole attacks and outline their strengths and weaknesses. [10], [11], and [12], have conducted a comprehensive set of simulations that illustrate the effects of a blackhole attack to the AODV routing protocol. In particular, the authors focus on the second part of the attack

(i.e., packet drop) and evaluate its impact to the packet delivery rate of the network, the end-to-end delay, as well as the throughput, under various mobility scenarios. However, none of these works provide any insights regarding the first step of the attack, the related routing parameters that are exploited by a malicious node, or how these parameters affect the attack itself (i.e., such as the percentage of routes won by a malicious node).

A variety of detection mechanisms for blackhole attacks in AODV also exists in the literature and even though we provide an evaluation of the most recently proposed solutions, a comprehensive analysis of all the related literature requires an extensive review, which is outside the scope of this paper. In [13], a distributed cooperative mechanism (DCM) is proposed to resolve blackhole attacks, by monitoring data packets transmitted by neighboring nodes. If a node has not routed any data packets during a fixed time-threshold, then the monitoring node will transmit a "test packet" through the suspicious node, destined for another cooperating detection node. If the later receives the "test packet," then the suspicious node is legitimate; otherwise, it is considered malicious. The primary disadvantage of this scheme is that malicious nodes may attempt to exploit this mechanism, by analyzing the duration of time before a malicious node is detected (i.e., estimate the threshold value), and subsequently, the routing of at least one packet within this time-frame (i.e., selective drop).

To address the limitation of [13], [14] proposes the use of a dynamically updated normal profile. In this scheme, the normal profile is updated dynamically, using monitored data collected during a period of time in which no malicious behavior was detected. It utilizes a support vector machine classifier (SVM) for detecting an attack by monitoring the delay between data transmissions. Although the use of dynamic profiles may reduce the rate of false positives in volatile networks; on the other hand, by relying on data transmissions for detection, attacks in which data packets are selectively dropped, remain undetected.

In [15], the authors propose a mechanism to detect blackhole attacks by checking if the SQN of a RREP message is higher than a dynamic threshold value, which is an indication of a blackhole attack. The value of the threshold is updated by calculating the difference between the SQNs of the RREP message and the average of the previously received SQNs. However, in case of high mobility, the exchanged routing information is greatly increased (i.e., caused by link breakages), resulting in an unexpected increase in the SQNs of control packets, and thus, leading to considerably high false alarms. Moreover, the proposed solution requires many significant modifications to the AODV protocol.

In [16], the authors propose a reputation scheme called Prevention of Cooperative Black-Hole Attacks (i.e., PCBHA). In this scheme, each node maintains reputation scores for the other nodes of the network and when a route is required, the source node selects the route that includes intermediate nodes with the highest reputation scores. The carried out simulation results show that the performance of the AODV protocol is not deteriorated, considerably, using

the proposed solution. However, the reputation information exchanged between nodes results in additional communication overhead and the proposed scheme is vulnerable to byzantine attacks, since a colluding group of malicious nodes may exploit the proposed scheme by providing fake reputation values that are high.

A modified version of AODV, referred as the Gratuitous-AODV (i.e., GAODV), has been proposed in [17], in order to address the issue of blackhole attacks. In GAODV, when a source node receives a RREP from an intermediate node, it sends a verification message to the destination node. The latter should also provide an acknowledgment message to the source node. If the source node does not receive the acknowledgment, then the intermediate node is considered malicious and thus, the advertised routing path is not used. However, the functionality of GAODV requires extensive modifications to the original AODV protocol, raising compatibility issues and it introduces considerable delay in the route discovery process.

Finally, in [18], the authors propose a detection mechanism called the Anti-Blackhole Mechanism (i.e., ABM), which captures both RREQ and their corresponding RREP messages and, subsequently, estimates the difference between the two. When this difference exceeds a predefined threshold, an alarm is raised informing all nodes on the network to cooperatively isolate the malicious node. ABM requires each node to run in promiscuous mode in order to capture, store, and, subsequently, process the RREQ and RREP messages within their radio range. Consequently, monitoring nodes are hindered with computational and storage overheads, as well as increased energy consumption. In addition, during the collection of captured traffic, malicious activities are not detected (i.e., non-real-time detection). The functionality of ABM also requires the operation of a modified version of the AODV protocol (i.e., MAODV), raising compatibility issues with the AODV protocol.

In summary, existing detection mechanisms are limited in the sense that their deployment requires significant modifications to the AODV protocol [17][18], while some of the proposed solutions add considerable performance delays and communication overheads [13][16][17]. Even more importantly, the majority of these mechanisms attempt to resolve if a blackhole attack takes place, based only on the second step of the attack (i.e., packet drop) [13][14][16][17]. Thus, they do not completely mitigate the attack (since detection can only be achieved after the malicious node wins the route discovery process), and they are effective, only, when the malicious node indiscriminately drops all of the forwarded traffic. On the other hand, our proposed detection mechanism is capable of detecting a blackhole attack during its first step (i.e., during the exploitation of the route discovery process), limiting the ability of a malicious node to drop packets, and thus, induce damage onto the network. Furthermore, by disassociating the detection of an attack from packet drop monitoring, the proposed detection mechanism is capable of detecting not only the blackhole attack but also the greyhole, in which a malicious node selectively drops packets, in order to avoid detection, in which a malicious

node might selectively drop packets, in order to avoid detection. Finally, the proposed mechanism alleviates any associated communication overheads and does not require any modifications to the existing AODV routing protocol.

3 A comprehensive analysis of the blackhole attack

In this section, we investigate the exploitation of the route discovery process during a "reactive" blackhole attack, and, identify a new critical attack parameter (i.e., blackhole intensity). Based on this new parameter, we then perform a set of simulations to determine the impact of SQN exploitation on the performance of blackhole attacks, as well as the optimal blackhole intensity parameter values that can be used by a malicious node, in order to maximize the outcome of an attack. The performed simulations also take into account the impact of various network conditions such as node's mobility, variable density of nodes, and variable traffic.

3.1 The blackhole intensity parameter

In a blackhole attack, the objective of a malicious node is to attract as much traffic as possible, in order to maximize the number of packets that can be dropped, when legitimate source nodes transmit data. This is achieved during the first step of the attack, in which the malicious node provides a fake SQN (i.e., denoted as $SQN_{malicious}$) greater than all other SQN values provided by legitimate nodes, and, thus, wins all the received route requests. This can be clearly seen in the example of section 2.2; at step b. Furthermore, the parameter $SQN_{malicious}$ affects not only the source node that initiated the route request, but also all intermediate nodes (such as node I_1 in the example) that stored this parameter in their routing tables. However, the malicious node cannot discern what the current values are for the SQNs of other nodes. Thus, it must increment the SQN with a value high enough, to overcome legitimate nodes competing for the route discovery process (i.e., nodes I_2 and I_3 in the example). We define this increment as the *blackhole intensity* parameter or parameter L for short. Let $SQN_{malicious}$ be equal to the destination SQN in the RREP message (i.e., SQN_{dst_node}), incremented by a value L. That is,

$$SQN_{malicious} = SQN_{dst\_node} + L, \qquad L \geq 0 \tag{1}$$

Evidently, the value of the destination SQN_{dst_node} in the RREP message will be selected by the attacker so that to be the highest between the destination SQN received in the RREQ message and the one stored in its routing table (if it has a stored one). In the example presented in section 2.2, the malicious node increments SQN_{dst_node} by a blackhole intensity parameter value equal to 1000. The blackhole intensity parameter plays a crucial role to the success of the attack, because it determines whether or not the malicious node will win a route request, and thus, attract traffic. However, there is no indication as to what values this parameter should hold, and how this affects the outcome of the attack. For example, if the malicious node selects a relatively "small" value for L, then the malicious node might not win all of the route requests.

This result might be further exacerbated under different network conditions. In particular, a higher number of traffic will lead to higher SQN values for competing legitimate nodes, and thus, even less route request wins for the malicious node. On the other hand, selecting a relatively "high" value for L may be counterproductive, because after some threshold, the malicious node will be wining all of the received route requests, and thus, higher values of L yield no further benefit. Moreover, since our goal is to utilize SQNs for detection, there is an additional incentive for the attacker to use the lowest values of L possible, in order to hinder the ability of a detection mechanism to distinguish its malicious activity. In order to accurately quantify the impact of the blackhole intensity parameter, we have conducted a comprehensive set of simulations that are presented in the following section.

3.2 Evaluating the impact of the blackhole intensity parameter through simulations

In this section, we present the numerical results of an extensive set of simulations, which exemplify the quantitative relation between SQNs and blackhole attacks. More particularly, we utilize the blackhole intensity parameter in order to ascertain: (i) the impact of SQN exploitation on the performance of blackhole attacks (i.e., the percentage of routes won (PRW) during the route request process and the percentage of packets dropped, by the malicious node); (ii) the optimal blackhole intensity parameter values that can be used by a malicious node to maximize the effectiveness of the attack (i.e., the minimum values of the blackhole intensity parameter that yield the maximum percentage of route requests won by the malicious node); and (iii) assess how all the above are affected by various network conditions such as node's mobility, variable density of nodes and variable traffic.

To quantify the performance and facilitate the evaluation of the blackhole attack, we have assessed: (i) the PRW and (ii) the packet delivery ratio (PDR), both as a function of the blackhole intensity parameter. PRW indicates the route requests that the malicious node managed to win, as a ratio of the total RREQ messages received by it. PDR, on the other hand, indicates the impact of the attack in the normal operation of the network, i.e., the percentage of packets that manage to be delivered, as a ratio of the packets that were transmitted on the network. The simulation parameters were selected based on the parameters used by the related work [13-18], in order to generate similar experimental conditions. The simulations were performed using the ns-3 network simulator, version 3.20, which was tested and validated after the installation. The underlying network topology was constructed by, randomly, placing 25, 50, 75, or 100 nodes in an area of 1000m x 1000m, where nodes established links if they were in a radio range of 100m or less. Network traffic was generated by randomly selected constant bit rate (CBR) source nodes, which transmitted 512-byte data packets at a fixed rate of 4 packets per second, at a randomly selected destination node. In the related literature, the number of

source nodes is typically fixed to 10 randomly selected source nodes. However, in this set of simulations, we have extended this parameter to either 25%, 50%, or 75% of the total number of network nodes, in order to take into account a larger set of network conditions. The blackhole attack was performed by one malicious node, which was selected randomly (i.e., uniformly distributed) from the set of nodes not designated as either source or destination. Nodes' mobility was simulated using the random waypoint mobility model and the speed of mobile nodes ranged from 0 to 10 m/s. Finally, each simulation experiment was executed 10 times and the results were averaged. Performing more than 10 runs did not provide any significant statistical deviation in the average value. Table 1 displays a summary of the simulation parameters.

Table 1: Simulation parameters

Simulation parameters	***Value***
Number of nodes	25, 50, 75, 100
Simulation area	1000 m x 1000 m
Radio range	100 m
Mobility model	Random waypoint
Nodes' mobility	Ranged from 0 to 10 m/s
Channel capacity	2Mbps
Traffic type	CBR
Traffic flow	10 source nodes or 25%, 50%, 75% of network nodes
CBR packet size	512 bytes
Number of malicious nodes	1
Values of L	0-1000

Figure 2 presents the PRW by a malicious node as a function of blackhole intensity, under various mobility scenarios. The goals for this experiment were to identify: (i) if the blackhole intensity parameter has any impact to the route discovery process, and (ii), what that impact is, under a variety of mobility scenarios (i.e., average node speed of 0, 2, 4, 6, 8, and 10 m/s). Since PRW indicates the route requests that the malicious node managed to win, as a ratio of the total RREQ messages received by it, the number of nodes and the traffic flow have no impact on PRW, and thus, both parameters remained fixed (i.e., the number of nodes was equal to 50, while the traffic flow was equal to 50% of network nodes).

When the blackhole intensity value was equal to 0, the performance of the attack was close to a simple packet drop, since the malicious node did not advertise a "fresher" route (it did however advertise a route to a destination that it does not possess). More specifically, in the absence of mobility, the malicious node managed to win 20% of the received route requests, while at the presence of higher node's speed this percentage dropped, reaching 7%, for an average node's speed of 10 m/s. Increasing the blackhole intensity value resulted in an increase in the percentage of routes won by the malicious node. In particular, for a blackhole intensity value equal to 1, the PRW ranged from 21% to 9%, while for blackhole intensity value equal to 5, the PWR ranged from 69% to 48%. Finally, for blackhole intensity values above 11, the PRW was 100%. Based on these results, we can make the following observations. First off, increasing the blackhole intensity parameter leads to an increase in PRW. This outcome can be attributed to the fact that a higher blackhole intensity parameter increases the chances for a malicious node to overcome the destination SQN values of any other legitimate node competing for the same route request (i.e., since the malicious node is not aware of the exact destination SQN values stored by other nodes). Another observation that can be drawn from these results is that, when nodes' mobility increases, the percentage of routes won by the malicious node drops. This is due to the link breakage, which: (i) obstructs some of the malicious node's route reply messages to reach their intended destination and (ii) leads to an increase in the SQN of legitimate nodes, enabling them to win some of the routes to the destinations.

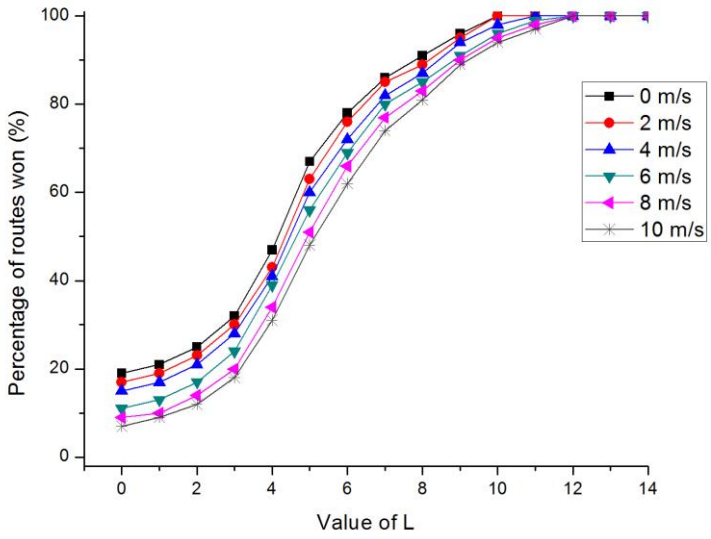


Figure 2: PRW by the malicious node as a function of blackhole intensity

Having established a relationship between the blackhole intensity parameter and the percentage of routes won by the malicious node, we now extend our study to the second part of the attack (i.e., the packet drop). Figure 3 presents the PDR as a function of blackhole intensity,

under various mobility scenarios (i.e., average node speed of 0, 2, 4, 6, 8, and 10 m/s). The goals of this experiment were to identify: (i) if the blackhole intensity value has any impact to the PDR of the network, and (ii), what this impact is, under various mobility scenarios. The other parameters remained fixed (i.e., the number of nodes was equal to 50, while the traffic flow was equal to 50% of network nodes).

According to Figure 3, in the absence of mobility, the PDR dropped to 0%, for blackhole intensity values of 10 and above. Thus, a blackhole attack managed to win all of the route requests, when it utilized a blackhole intensity value higher than 10. Moreover, in the case of stable nodes (i.e., no mobility), the links between source and destination nodes remained constant (i.e., there was no link breakage), and, as a result, the malicious node dropped all of the transmitted traffic. In the case of mobile nodes, and in particular, when the average node's speed was 2 m/s, PDR dropped to 57% for blackhole intensity values of 11 and above. When the average node's speed increased to 4 m/s, the PDR dropped to 62% for blackhole intensity values of 11 and above. At an average node speed of 6 m/s, the PDR dropped to 68% for blackhole intensity values of 12 and above. Finally, for an average node speed of 8 m/s and 10 m/s, the PDR dropped to 75% and 84% respectively, for blackhole intensity values of 12 and above. Based on these results, we can make the following observation: in the presence of mobility, the impact of the attack to the PDR decreases. This outcome is once again attributed to the link breakage that is caused by the mobility of nodes. In particular, when link breakage occurs, the malicious node will often lose connectivity with traffic flows that it was previously attacking (i.e., won the route request for that traffic flow), thus limiting the amount of packets it can drop.

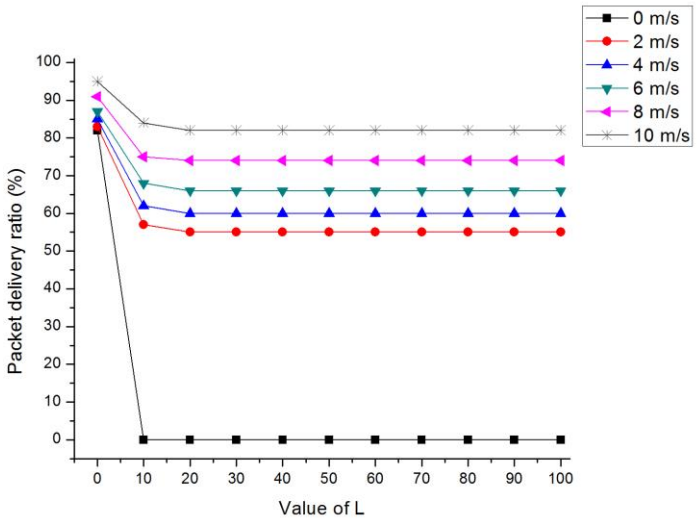


Figure 3: PDR as a function of blackhole intensity with variable network speed and a fixed traffic flow of 50%.

Figure 4 presents the PDR as a function of blackhole intensity, under various traffic flow scenarios (i.e., the percentage of nodes generating traffic was set to 25%, 50%, or 75% of nodes). The goal of this experiment was to identify if a variety of traffic flow conditions may also affect the impact of the blackhole intensity value to the PDR. The other parameters remained fixed (i.e., the number of nodes was equal to 50, while the average nodes' mobility was equal to 5 m/s). When the percentage of nodes generating traffic was at 25%, the PDR dropped to 76% for blackhole intensity values of 11 and above. When the percentage of traffic flows increased to 50%, the PDR dropped to 64% for blackhole intensity values of 11 and above. Finally, when the percentage of traffic flows increased to 75%, the PDR dropped to 52% for blackhole intensity values of 11 and above. Based on these results, we can deduce that increasing the percentage of traffic flows resulted in a higher percentage of packets dropped by the malicious node (i.e., the PDR dropped). This outcome is attributed to the fact that a higher number of traffic flows led to more route requests reaching the malicious node, which in turn, was capable of advertising the spurious path more often, and thus, drop a higher percentage of packets.

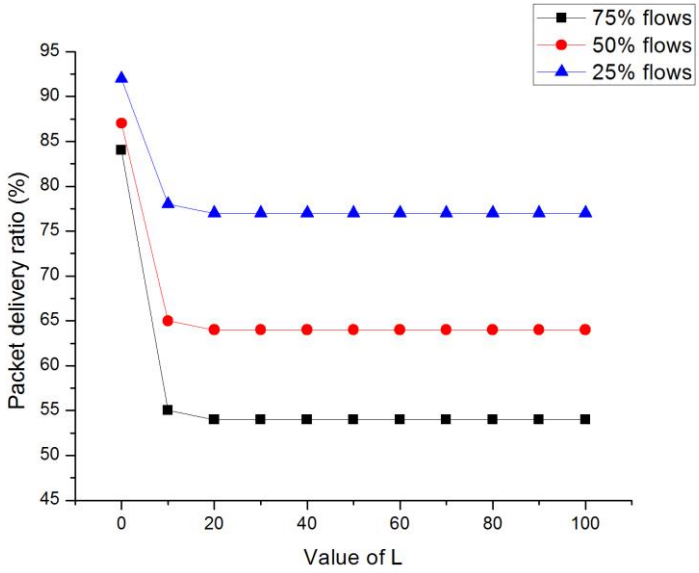


Figure 4: PDR as a function of blackhole intensity with variable number of traffic flows and a fixed average network speed of 5 m/s.

Figure 5 presents the PDR as a function of blackhole intensity, under various number of nodes scenarios (i.e., the number of nodes was set equal to 25, 50, 75, and 100). The goal of this experiment was to identify if a variety in the number of nodes may also affect the impact of the blackhole intensity value to the PDR. The other parameters remained fixed (i.e., traffic flow was equal to 10 source nodes, while the average nodes' mobility was equal to 5 m/s). According to Figure 5, in a network of 25 nodes, the PDR was equal to 59%, for blackhole intensity values of 11 and above. When the number of nodes increased to 50, the PDR also

increased to 73%, for blackhole intensity values of 11 and above. In a network of 75 nodes, the PDR further increased to 84%, for blackhole intensity values of 11 and above. Finally, when the number of nodes increased to 100, the PDR increased to 91% for blackhole intensity values of 11 and above. Based on these results, we can deduce that increasing the number of nodes resulted in a lower percentage of packets dropped by the malicious node (i.e., the PDR increased). This outcome is attributed to the fact that a higher number of nodes led to less route requests reaching the malicious node, which in turn, was capable of advertising the spurious path less often, and thus, drop a smaller percentage of packets.

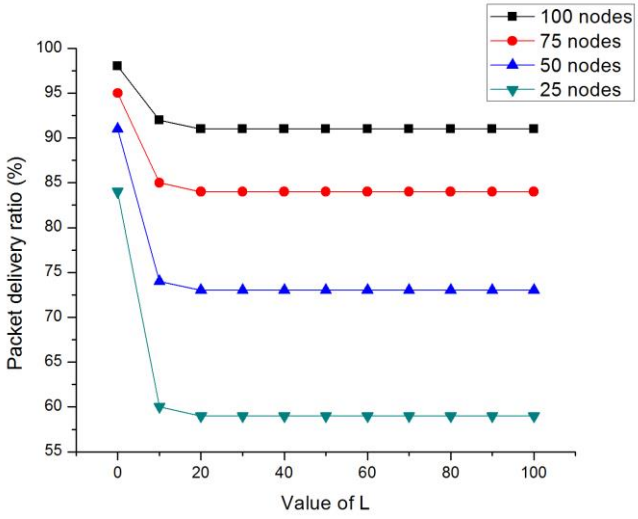


Figure 5: PDR as a function of blackhole intensity with variable number of nodes and a fixed average network speed of 5 m/s.

3.3 Analysis of the blackhole intensity parameter simulations

By examining the results of the simulations presented in section 3.2, we observe that the newly identified blackhole intensity parameter does indeed play a key role in the outcome of the attack. As illustrated in Figure 2, when the malicious node selects a higher value for the blackhole intensity parameter, it increases its chances to win a received route request. This is attributed to the fact that a higher blackhole intensity parameter increases the chance of the malicious node to overcome the SQN values of any other legitimate node competing for the same route request (i.e., since the malicious node is not aware of the exact SQN values stored by other nodes). Moreover, as we can see in Figures 3 and 4, the immediate consequence of the malicious node winning a greater percentage of route requests is a decrease in the PDR of the source nodes. By winning more route requests, the malicious node manages to attract a greater percentage of network traffic (i.e., since more source nodes select it as a path to their destination, even if no such path exists) and thus, it is capable of dropping a larger percentage of packets.

Based on the simulation results, we can also deduce that there is an optimal range of blackhole intensity parameter values that can be used by an attacker. As illustrated in Figure 2, the malicious node manages to win all of the received route requests when using blackhole intensity parameter values ranging between 10 and 12 (depending on the mobility of nodes). Using higher blackhole intensity parameter values becomes inconsequential, and will not provide an additional benefit to the attack (i.e., a higher percentage of routes won). On the other hand, when the malicious node uses a blackhole intensity parameter value lower than 10, the percentage of route requests won begins to decline. Thus, it can be concluded that a blackhole attack does not require an arbitrary high increment of the SQN parameter to be successful, since, in most cases, an increment of 10 is sufficient. Furthermore, the optimal blackhole intensity parameter is highly reliant on network conditions (i.e., average node speed and number of network flows). According to the results of the simulations, a higher average node's speed results in a higher PDR, which means that the malicious node is dropping a smaller percentage of packets. On the other hand, a higher number of network flows results in a PDR decrease, which means that the malicious node is dropping a higher percentage of packets. An attacker that aims at utilizing the lowest possible values for the blackhole intensity parameter (for example, to avoid detection), may attempt to assess the network conditions by monitoring the rate of path breakage with other nodes, as well as the rate of received packets, in order to estimate the speed of nodes and rate of flows and adjust the blackhole intensity threshold, accordingly.

The qualitative and quantitative relation between SQNs and blackhole attacks leads to the conclusion that the detection of blackhole attacks can be achieved by monitoring the behavior of SQNs. The advantages of such an approach are twofold: first of, blackhole attacks can be detected in real or near-real time, since this approach alleviates the need to wait until the malicious node begins dropping packets in order to detect the malicious behavior. As a result, the attack may be detected before the malicious node is capable of dropping any packets, thus limiting the damages that are induced onto the network. Secondly, a detection mechanism that relies on the behavior of SQNs instead of packet drops as a monitoring feature is capable of detecting a greater range of blackhole attacks. For example, a more intelligent variation of a blackhole attack (i.e., greyhole) might only selectively drop packets, in order to avoid detection. Current detection mechanisms that rely on monitoring packet drops for detection are ineffective against such attack variations.

4 CUSUM detection mechanism

In this section, we analyze and evaluate a novel blackhole detection mechanism that is capable of detecting blackhole attacks during their first step. Particularly, in section 4.1; we provide an architectural overview of the proposed detection mechanism, in section 4.2; we identify the

computational overhead associated with the operation of the proposed mechanism, while in section 4.3; we comparatively evaluate the performance of the proposed mechanism through an extensive set of simulations. The proposed mechanism uses a non-parametric version of the Cumulative Sum (CUSUM) test [19], with the goal of detecting abrupt changes in the normal behavior of SQNs, caused by the occurrence of blackhole attacks. Two variants of this mechanism are presented, depending on the type of threshold used (i.e., static or dynamic). The CUSUM test is a suitable solution for infrastructure-less networks, since, it does not impose significant computational overheads [20][21], meaning that the performance of the AODV protocol is not deteriorated. Moreover, it is insensitive to traffic patterns with unknown distribution, making the detection mechanism generally applicable, regardless of the employed application-layer protocols. Another advantage of using the CUSUM test is related to the fact that, given an appropriate threshold value, it detects the attack at the earliest possible time while maintaining a low percentage of false positives. It is evident that a fast detection mitigates the impact of blackhole attacks, because it limits the ability of an attacker to drop packets.

4.1 Architecture of the proposed detection mechanism

In the proposed scheme, each network node executes an instance of the detection mechanism, which relies solely on local audit data (i.e., there is no cooperation between nodes). Each of these instances, can be implemented at the application or routing layer of a device, alleviating the need for any AODV protocol modifications. During their execution, they passively monitor the SQN parameter values stored in the nodes' routing table, and, at predefined time intervals, run the CUSUM test, in order to determine if a blackhole attack takes place. More specifically, in case of a Linux based device, we have identified three different implementation options [28],[29]: i) sniffing, in which the node will promiscuously sniff all incoming packets on a network interface (the code to perform sniffing is built into the kernel and is available to user-space programs by using the Packet Capture Library (libpcap)); ii) kernel modifications, using either patches (low portability – low complexity solution) or recompilation of the whole kernel (high portability – high complexity solution); iii) Netfilter, which is a packet filtering framework implemented as a set of hooks at well-defined places in the Linux TCP/IP networking stack. The CUSUM test is a change point detection algorithm, which evaluates the statistical distribution of SQNs prior to change and after, and subsequently, raises an alarm if the difference between the two exceeds some threshold. The later can be either dynamic (i.e., dynamic threshold CUSUM) or static (i.e., standard CUSUM). In this analysis, both threshold variants are elaborated, and, subsequently (see section 4.3.2), the most suitable threshold mechanism is selected, by comparatively evaluating the detection accuracy and the rate of false positives between the two. The detection mechanism calculates the statistical distribution of SQNs based on the monitoring feature $SQN_{total\_rate\_i}(t)$ (see eq. 2). Formally, for some node i

executing an instance of the detection mechanism, we define this monitoring feature as the rate of increase for the sum of the SQNs included in the node's i routing table:

$$SQN_{total\_rate\_i}(t) = \frac{\left(\sum_{j=1}^{K} SQN_{routing\_table\_i\_j}(t)\right) + SQN_i(t)}{t} \qquad (2),$$

where $SQN_{routing\_table\_i\_j}(t)$ is the SQN value at time t of node j stored in the routing table of node i. K is the total number of entries in the routing table of node i, while $SQN_i(t)$ is the value of SQN of node i at time t.

At network initialisation, the CUSUM algorithm requires an initial statistical distribution of SQNs to compare to. As a result, two phases are incorporated into the detection mechanism, a training phase and a normal phase. We assume that during training, no attack takes place (i.e., training can be performed in a controlled environment), while during the normal phase, any node on the network can perform a blackhole attack. Furthermore, in both phases, the CUSUM algorithm is executed at a predefined, time interval. Since the detection of an attack requires the execution of the CUSUM algorithm, this time interval represents the detection time of the proposed mechanism. Therefore, it would seem practical to keep the time interval at the lowest possible value so that attacks are resolved quickly. However, this interval has an associated tradeoff: lower values produce more frequent executions of the detection mechanism, and, consequently, higher induced overhead. Larger values, on the other hand, may lead to: (a) the calibration of an outdated threshold value, resulting in a higher percentage of false positives, and (b) a greater percentage of packets dropped by the malicious node. Thus, the most optimal time interval is the largest possible value that produces the least amount of false positives and packets dropped. In section 4.3.1, through simulations, we identify the most optimal time interval value.

4.1.1 Training phase

During the training phase, at each time interval, the CUSUM algorithm first calculates a random sequence X_n which we define as the difference between two successive sampling values of the monitoring feature $SQN_{total\_rate\_i}(t)$. That is,

$$X_n = SQN_{total\_rate\_i}(n) - SQN_{total\_rate\_i}(n-1), \qquad X_0 = 0 \qquad (3).$$

Next, the CUSUM test transforms X_n to another random sequence Z_n such as:

$$Z_n = X_n - C, \quad C \in R \qquad (4),$$

where C is a constant variable that is equal to the upper bound of the mean value $E[X_n]$. The CUSUM algorithm also requires the calculation of a random sequence Y_n that represents the cumulative sum of the positive values of Z_n. Y_n is defined as the maximum value between zero and $Y_{n-1} + Z_n$. That is:

$$Y_n = max(0, Y_{n-1} + Z_n), \text{ where } n \in \mathrm{N} \text{ and } Y_0 = 0 \quad (5).$$

The value of the threshold N is computed at the end of the training phase by each node. Its value is equal to the mean value of the n samples of X_n. That is,

$$N = E[X_n] \quad (6).$$

The selection of threshold *N* regulates the following intrinsic tradeoff: having a relatively "small" threshold may lead to a high percentage of false positives, since even legitimate increases in the statistical distribution of SQNs will lead to false alarms, while, on the other hand, having a relatively "high" threshold may lead to false negatives, since increments to the SQN by a malicious node may not exceed the threshold, and, therefore, the attack will not be detected. We have based the selection of threshold *N* on previous literature [26][27], in which it yielded the most optimal results in terms of false positives/negatives. Furthermore, in section 4.3.2, through simulations, we assess the performance of our proposed CUSUM threshold variants based on their detection accuracy and false positive ratio.

4.1.2 Normal phase

During the normal phase, at each time interval, the CUSUM algorithm calculates all three random sequences X_n, Z_n, Y_n. It then uses the random sequence Y_n and the threshold N to detect blackhole attacks. In particular, the detection is based on the following simple rule: if at any time interval n, the random sequence Y_n exceeds the threshold N (i.e., $Y_n > N$), then a blackhole attack is detected and an alarm is raised to inform other nodes on the network. Finally, in the dynamic threshold variant of CUSUM, for each time interval in which an attack is not detected, the threshold N is also recalculated, to a value equal to the mean of X_n, X_{n-1} (7). Table 2 summarises the operation of the CUSUM algorithm during both phases.

$$N = E[X_n, X_{n-1}] \quad (7).$$

Table 2: Pseudocode of the CUSUM algorithm

CUSUM Algorithm
Input1: K // Number of routing table entries
Input2: *is_dynamic* // Boolean indicating the type of CUSUM (if TRUE then CUSUM is dynamic)
1: *set* $Y_0 = 0,\ n = 1$;
2: **while** Training
3: *compute* X_n, C, Z_n;
4: **if** $Y_{n-1} + Z_n > 0$
5: **then** $Y_n = Y_{n-1} + Z_n$;
6: **else** $Y_n = 0$;

```
7:  compute N;
9:  while Detection
10:     compute X_n, C, Z_n;
11:     if Y_{n-1} + Z_n > 0
12:        then Y_n = Y_{n-1} + Z_n;
13:     else Y_n = 0;
8:      if  Y_n > N
9:         then raise an alarm
10:     else
11:        if is_dynamic = TRUE
12:           then   compute N;
13:     n = n + 1;
```

4.2 Computational cost of the proposed detection mechanism

In this section, we quantify the computational cost for both variants of the proposed detection mechanism (i.e., standard and dynamic threshold CUSUM). For the purposes of this evaluation, first we determine the number of arithmetic operations, as well as the maximum number of comparisons performed by the detection mechanism, and then, we transform these operations into the equivalent number of CPU instructions. The later depends on the target CPU architecture. In this analysis, we assume that the target architecture is an ARM CPU that implements the thumb-2 instruction set [22]. This is a widely adopted architecture most common in mobile and embedded devices. A similar analysis can be performed for Intel's x86 CPU architecture [23]. The computational cost induced by either of the two CUSUM variants can be divided into two parts, the cost induced during (i) the training phase and (ii) the normal phase. Since the duration of these two phases is not fixed, we will concentrate this analysis on the computational cost induced at each time interval.

During the training phase, the two CUSUM variants have no differences, and thus, induce the same computational costs. More specifically, at each time interval n during training, each node must compute the three random sequences X_n, Z_n, Y_n and the variable C. According to equation (3), presented in section 4.1.1, the random sequence X_n requires the computation of the monitoring feature SQN_{total_rate}. For K number of routing table entries, the computation of SQN_{total_rate} requires K + 1 additions and 1 division, as defined by equation (2). Consequently, the computation of X_n requires K + 1 additions, 1 division, and 1 subtraction. The computation of the random sequences Z_n and Y_n that is presented in equations (4) and (5), requires 1 subtraction and 1 addition, respectively. Finally, the computation of parameter C requires 1 comparison, based on the parameter's definition in section 4.1.1. Thus, in total, at each time interval, the training phase requires K + 2 additions, 2 subtractions, 1 division, and 1 comparison. At the end of the training phase a one-time computation of the threshold N also

takes place, which requires n – 1 additions (i.e., where n is the current time interval) and 1 division, as illustrated by equation (6).

During the detection phase that is presented in section 4.1.2, in the standard CUSUM variant, for each time interval n every node must compute the three random sequences X_n, Z_n, Y_n, the variable C, and test if the random sequence Y_n exceeds the threshold N. Thus, in total, it requires K + 2 additions, 2 subtractions, 1 division, and 2 comparisons. On the other hand, the dynamic threshold CUSUM must compute the three random sequences X_n, Z_n, Y_n, the variable C, test if the random sequence Y_n exceeds the threshold N, and also compute a new threshold N, defined by equation (7), thus requiring K + 3 additions (i.e., where K is the number of routing table entries), 2 subtractions, 2 divisions, and 2 comparisons.

Having established the number of operations performed by the two CUSUM variants, we now estimate the equivalent cost in CPU instructions. An ARM processor requires 1 CPU instruction to perform either an arithmetic operation or a comparison between two numerical values stored in the CPU's registers [22]. Fetching the two values into the registers and storing the result back into memory requires 3 additional CPU instructions. Thus, each operation is estimated at a cost of 4 CPU instructions [22]. Based on the above, at each time interval, the CPU instructions required during training are: 4K + 24, where K is the number of routing table entries and n is the current time interval. At the end of the training phase, the computation of threshold N requires 4n CPU instructions. One the other hand, during the detection phase, standard CUSUM requires 4K + 28 CPU instructions, while the dynamic threshold variant requires 4K + 36 CPU instructions. Thus, during detection (i.e., normal phase), both CUSUM variants are characterized by linear computational complexity, for an input of K route table entries. Assuming, at the worst case that each node on the network possesses routing table entries for all other network nodes. Then, for a network of 100 nodes (i.e., the largest network size we evaluated) K will be equal to 99, since each node's routing table will have 99 entries (i.e., one route for all other nodes on the network, excluding itself). As a result, the standard and dynamic threshold variants of CUSUM will require 424 and 432 CPU instructions per time interval n, respectively. The computational capacity of an embedded microcontroller such as ARM's Cortex M4, designed for low power applications (including sensors and IoT devices) is 100 million instructions per second (MIPS) [24]. Therefore, the computational overhead produced by the proposed detection mechanism has negligible impact on the computational capacity of a modern embedded processor. This exemplified computational cost is significantly lower than the cost induced by other detection mechanisms present in the literature (a detailed evaluation is presented in section 4.3.4).

Table 3: Computational cost in CPU instructions for each CUSUM variant

CUSUM variant	Training phase (total cost)	Normal phase (cost per time interval n)
Standard CUSUM	$(4K + 24)n + 4n$	$4K + 28$
Dynamic threshold CUSUM	$(4K + 24)n + 4n$	$4K + 36$

4.3 Experimental evaluation

In this section, a prototype of the proposed detection mechanism is evaluated through simulations. The objectives of these simulations were to: (i) identify the most optimal value for the monitoring time interval (section 4.3.1); (ii) compare the two CUSUM variants (i.e., dynamic threshold and standard) and identify the most suitable threshold mechanism (section 4.3.2); (iii) evaluate the performance of the proposed detection mechanism in relation to the blackhole intensity parameter (section 4.3.3); and (iv) compare the performance of the proposed detection mechanism to other security mechanisms designed for the detection of the blackhole attack (section 4.3.4). Once again, the simulations were performed using the ns-3 network simulator. We used the same simulation parameters as section 3.2, with the following variations. The amount of traffic flows has a fixed value equal to 10 randomly selected source nodes, or variable and equal to 25%, 50%, 75% of total network nodes. The value of 10 source nodes was selected, in order to provide results based on similar experimental conditions with the related literature. Moreover, a new parameter is introduced, namely the training period, which is set to 20000 seconds. Finally, in the first experiment of section 4.3.4, the number of malicious nodes ranged from 0 to 10, in order to assess the scalability of the evaluated detection mechanisms in the absence, or in the presence of one or more malicious nodes. Table 4 displays a summary of the simulation parameters.

4.3.1 Identifying the optimal time interval parameter value

In order to identify the optimal value for the monitoring time interval of the proposed detection mechanism, we evaluated the average false positive ratio and the average number of packets dropped by the attacker, as a function of the monitoring time interval parameter in seconds (see Table 5). Recall from section 4.1 that the optimal time interval is the largest possible value that produces the least amount of false positives and packets dropped. This experiment has been conducted under various mobility scenarios (i.e., the average node speed in the experiment ranged from 0 m/s to 10 m/s), different values for traffic flows (i.e., 25%, 50%, 75% of network nodes) and various values for number of nodes (i.e., 25, 50, 75, 100).

Table 4: Simulation parameters

Simulation parameters	*Value*
Number of nodes	25, 50, 75, 100
Simulation area	1000 m x 1000 m
Radio range	100 m
Training period	20000 sec
Mobility model	Random waypoint
Nodes' mobility	Ranged from 0 to 10 m/s
Channel capacity	2Mbps
Traffic type	CBR
Traffic flow	10 source nodes or 25%, 50%, 75% of network nodes
CBR packet size	512 bytes

The employment of a relatively "low" time interval value (i.e., 1 second) resulted in a average false positive ratio of 40%. This can be attributed to the fact that a very small time interval did not allow the detection mechanism to adjust the CUSUM threshold. On the other hand, the malicious node did not drop any packets. This outcome resulted from the fact that the source node was still awaiting for route replies and thus, the attack was detected before there was any data transmission. When the monitoring time interval value was between 2 and 4 seconds, the average false positive ratio dropped between 9% and 11%, while, once again, the attack was detected before any packets were dropped. Finally, monitoring time interval values equal to 5 seconds, or higher, led to an increase in the false positive ratio. More specifically, the average false positive ratio increased to 58% for a time interval value of 10 seconds. This behavior is attributed to the fact that in a volatile network, higher monitoring time interval values result in CUSUM calibrating an outdated threshold value. Moreover, the malicious node was now able to drop packets before it was detected (i.e., packet drop steadily increased to 193 packets in average for a monitoring time interval value of 10 seconds). Based on these results, the most optimal time interval value is 3 seconds, since it is highest possible time interval that yields the least average false positive ratio, while allowing for the detection of the attack before the malicious node has enough time to begin dropping packets.

Table 5: False positive ratio for different time intervals

Time interval (sec)	1	2	3	4	5	6	7	8	9	10
Average false positive ratio (%)	40	10	9	11	25	39	42	50	55	58
Average packets dropped by the attacker	0	0	0	0	12	39	78	114	157	193

4.3.2 Comparative evaluation of the two CUSUM threshold variants

In this set of experiments, we compare the two CUSUM threshold variants (i.e., standard and dynamic threshold) and identify which is the most suitable for infrastructure-less networks. This comparison, is based on the following metrics: (i) the detection accuracy as a function of the blackhole intensity parameter and (ii) the false positive ratio as a function of the average node's speed. The detection accuracy indicates the percentage of attacks detected, while the false positive ratio indicates the percentage of alarms that were not malicious behaviors.

Figure 6 presents the detection accuracy of the two CUSUM variants as a function of the blackhole intensity parameter. The goal of this experiment was to evaluate the ability of the two variants to detect the attack, regardless of the blackhole intensity value used by the malicious node. The experiment was conducted under fixed mobility, traffic flow, and number of nodes (i.e., 10 m/s, 10 source nodes, and 50 nodes, respectively) so that detection accuracy was only affected by the blackhole intensity parameter. Note that we have also performed experiments with different values for mobility, traffic flow and number of nodes that presented similar results, and for this reason we do not present them. Standard CUSUM was not capable of detecting the blackhole attack until blackhole intensity reached a value of 46 and above, while the dynamic threshold CUSUM yielded much better performance, as it was capable of detecting blackhole attacks for blackhole intensity values of 4 and above. From these results we can deduce that under the standard CUSUM, the malicious node was capable of winning all of the route requests, as long as it used a blackhole intensity value lower than 46. In contrast, during the deployment of the dynamic CUSUM, the malicious node could only win around 20% of the route requests (i.e., using blackhole intensity values lower than 4). Standard CUSUM reached its maximum detection accuracy (i.e., 98%) at a blackhole intensity value of 76, while the dynamic threshold CUSUM reached the same detection accuracy at a lower blackhole intensity value (i.e., 14). Beyond these values both variants produced the same results. The ability of the dynamic threshold CUSUM to detect a blackhole attack at a much lower blackhole intensity value than the standard CUSUM is attributed to the fact that the later utilizes a fixed threshold generated during training, while the former updates the threshold based on the latest sampling of SQN_{total_rate} and thus, maintains a lower, and much more accurate threshold value. However, the advantage of dynamic CUSUM comes with a tradeoff, common to the majority of detection mechanisms that rely on dynamic thresholds: if an attack is not

detected, then it may become part of the threshold during re-training, and thus, as long as the attacker uses the same blackhole intensity value, the behavior will remain undetected.

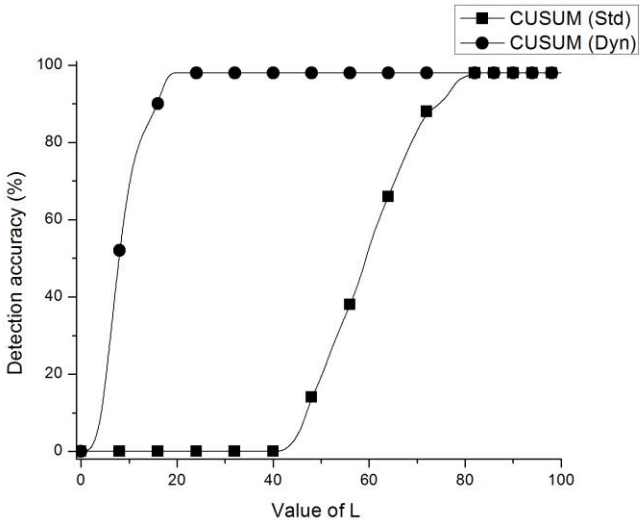


Figure 6: Detection accuracy as a function of the value of L

Figures 7 presents the false positive ratio for both CUSUM variants of the proposed detection mechanism as a function of the average node's speed; while Table 6 the percentage of traffic flows, respectively. The goal of these experiments was to assess how the two variants perform under high network volatility, and, in particular, when (i) the average nodes' speed and (ii) the percentage of traffic flows increase, or decrease, in relation to the values used in the initial training. Note that we have also performed experiments with different values for the number of nodes, which produced results comparable to Table 6.

In the first experiment (i.e., Figure 7), for both CUSUM variants initial training took place on a network with an average node's speed of 5 m/s, so that we can then assess the false positive ratio when mobility either drops, or increases. The number of nodes and traffic flow remained fixed (i.e., the number of nodes was equal to 50, while traffic flow was equal to 10 randomly selected nodes). Subsequently, we conducted 11 iterations of the experiments' second step, in which we increased the average node's speed by a factor of 1 m/s, beginning with an initial value of 0 m/s. The dynamic threshold CUSUM variant averaged a false positive ratio of 8%. On the other hand, the false positive ratio of the standard CUSUM variant increased both when the average speed decreased as well as when it increased. In particular, it reached 41% when nodes were static (i.e., average speed of 0 m/s) and 58% when the average node's mobility was 10 m/s. This outcome is attributed to the fact that the change in average network speed resulted in a different rate of increase for SQN_{total_rate} (i.e., caused by link breakages). The standard variant of CUSUM utilizes a static threshold value resulting from the training period in which the average node's speed was 5 m/sec and thus, its threshold was computed based on an outdated SQN_{total_rate}, while the dynamic threshold CUSUM variant is capable of adjusting

the threshold value and perform "retraining" when such network changes take place, as expected.

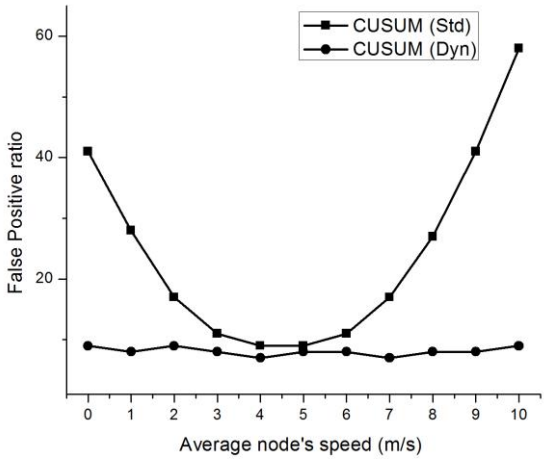


Figure 7: False positive ratio as a function of the average node's speed

In the second experiment (i.e., Table 6), initial training took place on a network with a traffic flow equal to 50 % of network nodes, so that we can then assess the false positive ratio when traffic flow either drops, or increases. The number of nodes and the average node's speed remained fixed (i.e., the number of nodes was equal to 50, while node's speed was equal to 5 m/s). Subsequently, we conducted 2 iterations of the experiments' second step, in which we adjusted the percentage of traffic flows to 25%, and 75%, respectively. Based on the simulation results, the dynamic threshold CUSUM variant averaged a false positive ratio of 8%. On the other hand, the false positive ratio of the standard CUSUM variant increased when a change in traffic flow occurred. In particular, it reached 22% when the percentage of traffic flows decreased (i.e., traffic flows equal to 25%), and 24% when the percentage of traffic flows increased to 75%. This outcome is attributed to the fact that the change in the number of traffic flows (i.e., from 50%, to either 25% or 75%) affected the exchanged routing information, resulting in an unexpected change of the SQN_{total_rate}, and thus, led to a considerably high percentage of false positives. Once again, the standard variant of CUSUM utilizes a static threshold value resulting from the training period in which the percentage of traffic flows was at 50%, and thus, its threshold was computed based on an outdated SQN_{total_rate}, while the dynamic threshold CUSUM variant is capable of adjusting the threshold value and perform "retraining" when changes in the percentage of traffic flows take place.

Table 6: False positive ratio under different traffic flow percentages for both CUSUM variants

Traffic flow	25% of network nodes	50% of network nodes	75% of network nodes
Standard CUSUM false positive ratio	22%	8%	24%
Dynamic threshold CUSUM false positive ratio	8%	8%	8%

Based on the experiments above, we conclude that the dynamic threshold variant of CUSUM outperforms the static variant, since it is capable of detecting blackhole attacks at lower blackhole intensity values (and thus, prevents them from wining the majority of route requests), while maintaining a constant false positive ratio, even under high network volatility. Furthermore, the computational cost between the two variants is insignificant, since both have to calculate the random sequences X_n, Z_n, Y_n, while the dynamic threshold variant has to perform one additional operation, for the re-calibration of its threshold, i.e., to compute the mean value between the previous threshold value and X_n. As a result, all of the subsequent experiments were performed using only the dynamic threshold variant of CUSUM.

4.3.3 CUSUM performance evaluation

In this set of experiments, we evaluate the performance of the proposed detection mechanism during a blackhole attack. To achieve this, we have derived PRW and PDR both as a function of the blackhole intensity parameter. Recall from section 3.2 that PRW indicates the route requests that the malicious node managed to win as a ratio of the total RREQ messages received by it, while PDR indicates the impact of the attack in the normal operation of the network, i.e., the percentage of packets that manage to be delivered, as a ratio of the packets that were transmitted on the network.

Figure 8 depicts the percentage of routes won by the malicious node as a function of the blackhole intensity parameter, while the dynamic threshold CUSUM is executing. The goal of this experiment was to evaluate the ability of the proposed mechanism to detect the blackhole attack, before the malicious node won the route request process, and under a variety of blackhole intensity values. The experiment was conducted using various mobility scenarios (i.e., the average node speed in the experiment ranged from 0 m/s to 10 m/s). Since the number of nodes and the traffic flow have no impact on PRW, both parameters have fixed values. That is, the number of nodes was equal to 50, while the traffic flow was equal to 10 randomly selected nodes.

As shown in Figure 8, when the blackhole intensity parameter was equal to 0, the performance of the attack was close to a simple packet drop attack, since the malicious node did not advertise a "fresher" route. In the absence of mobility, the malicious node managed to win 19% of the received RREQs, while at the presence of higher node's speed this percentage

dropped, reaching 7% for an average node's speed of 10 m/s. When the blackhole intensity increased to a value of up to 3, it resulted in an increase of the PRW by the malicious node. In particular, PRW ranged from 32% to 18% (for an average node's mobility of 0 and 10 m/s, respectively). This result is similar to our previous experiment (i.e., Figure 2), when the attack took place in the absence of any detection mechanism. However, in Figure 2, the PRW continued to increase as the value of the blackhole intensity value increased. On the contrary, as shown in Figure 8, for blackhole intensity values above 3, PRW becomes a decreasing function. That is, for a blackhole intensity value of 4, PRW ranged from 25% to 7%, for an average node's mobility of 0 m/s and 10 m/s, respectively. Moreover, for blackhole intensity values equal to 14 and above, PRW of the malicious node was 0% (for all the mobility scenarios). Therefore, from Figure 8 we can deduce that, when the PRW starts to decrease (i.e., for blackhole intensity values above 3), the dynamic threshold CUSUM starts the detection of the attack. On the other hand, in figure 2 the PRW increases for all blackhole intensity values, since there is no detection mechanism.

Based on these results, we can conclude that under the worst case scenario (i.e., absence of mobility and a blackhole intensity parameter value equal to 3), the malicious node will only be able to win 32% of the RREQs. Furthermore, when the attack is implemented using blackhole intensity parameter values equal to 5 or higher, the outcome is counter-productive for the malicious node, since the PRW is actually less than what the malicious node would gain if a blackhole intensity value of 0 was used (i.e., if the malicious node did not attempt to advertise a "fresher" route).

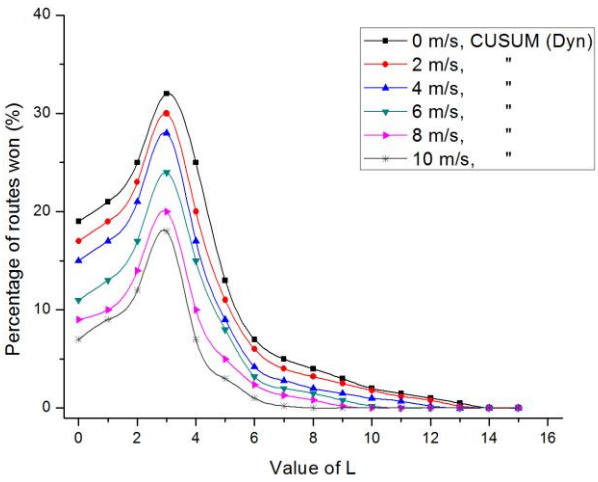


Figure 8: Percentage of route won as a function of the value of L with a fixed traffic flow of 10 source nodes

Figure 9 depicts the average PDR when a blackhole attack is performed using various values of the blackhole intensity parameter, with and without the execution of the dynamic

threshold CUSUM. The goal of this experiment was to identify the improvement of PDR in the network, when the proposed detection mechanism was introduced. To accomplish this, we are comparing the operation of the AODV protocol under a blackhole attack, both in the presence, and absence, of the proposed detection mechanism. The experiment was conducted under fixed mobility, traffic flow, and number of nodes (i.e., 10 m/s, 10 source nodes, and 50 nodes, respectively) so that the packet drop ratio was only affected by the blackhole intensity parameter. Similar experiments have been performed with differentiated parameters that produced the same results. When the blackhole intensity parameter was equal to 0, the PDR was similar i.e., 93% for both cases (i.e., with and without the execution of the detection mechanism). This outcome is attributed to the fact that even though the malicious node used a legitimate SQN value (i.e., recall from equation (1) that when the blackhole intensity parameter is equal to 0, then the value of $SQN_{malicious}$ is equal to SQN_{dst_node} it still advertised a route to a destination that it may not possess and dropped any packets that randomly traversed it. Therefore, in this scenario, the blackhole attack operated more as a simple packet drop attack. Both AODV and the detection mechanism exhibited similar results, for blackhole intensity values of up to 3, where the PDR under the execution of the detection mechanism reached its lowest value i.e., 86%. This decline in the PDR is attributed to the higher blackhole intensity parameter, which did not exceed CUSUM's detection threshold and, thus, enabled the malicious node of winning more route requests without being detected. At blackhole intensity values of 4 and above, the PDR without the execution of the detection mechanism continued to drop i.e., 83%, while in the second scenario (i.e., with the detection mechanism) it increased to 87%. This PDR increase (in the presence of the detection mechanism) indicates that the utilized blackhole intensity value exceeded CUSUM's threshold, resulting in the detection of the malicious node and the impediment of its ability to win route requests. Finally, in the absence of a detection mechanism, the PDR reached its lowest value i.e., 75%, at a blackhole intensity value of 9 and above, while in the second scenario (i.e., with the execution of the detection mechanism), the PDR reached 97% for blackhole intensity values of 14 and above. Based on these results, we can conclude that in the absence of a detection mechanism and as the value of the blackhole intensity parameter increases, the PDR drops i.e., the malicious node wins more routes and thus drops more packets. On the other hand, when the detection mechanism is deployed, it begins detecting the attack for blackhole intensity values of 4 and above. Since the attack is detected during the route discovery process, the attacker is impeded from winning the route request, and, thus, the PDR is increased. Therefore, it is counterproductive for an attacker to perform a blackhole attack in the presence of a detection mechanism, because using blackhole intensity parameter values higher than 4 pinpoint its presence, hindering its ability to drop packets.

Moreover, we elaborate on the detection time of the proposed mechanism. More specifically, first we define the detection time as the time duration between the attacker initiating the attack by broadcasting the fake SQN and the proposed mechanism detecting the blackhole attack. Please recall that the proposed mechanism (i.e., CUSUM algorithm) is executed periodically at predefined, time intervals. It is important also to notice that the proposed mechanism will either detect an ongoing attack or the attack will remain undetected until its end. Recall also that our proposed mechanism can detect attacks for blackhole intensity values greater than 3 (i.e., $L > 3$), while lower values for the blackhole intensity remain undetected (note however that for $L \leq 3$ the impact of the attack is negligible, since the malicious node will only win a small percentage of route requests – see section 4.3.3). Therefore, assuming a blackhole attack has been initiated by a malicious node, the attack will be detected at the next execution of the CUSUM algorithm. In other words, the time interval represents the maximum detection time of the proposed mechanism. As mentioned in section 4.3.1, after an experimental evalutaion we derived that the optimal value of the time interval in order to minimize false positives is 3 seconds. Thus, we deduce that the maximum detection time of the proposed mechanism is 3 seconds. On the other hand, the minimum detection time is the execution time of the CUSUM algorithm itself, assuming that the detection algorithm will be executed immediately after the malicious node initiates the blackhole attack (i.e., at the end of the time interval). Recall from section 4.2 the standard and dynamic threshold variants of CUSUM require 424 and 432 CPU instructions, respectively. Therefore, for an embedded microcontroller, capable of 100 million instructions per second, the execution time of the CUSUM algorithm (and consequently the minimum detection time) is in the order of a few microseconds.

Finally, we have estimated the PDR as a function of the average node's speed in the absence of any malicious node to measure the impact of the proposed mechanism to the normal operation of the network when no blackhole attacks take place. We conducted 6 iterations of this experiment, in each of which we increased the average node's speed by a factor of 2 m/s, beginning with an initial value of 0 m/s, up to a maximum of 10 m/s. The experiment was conducted under fixed traffic flow and number of nodes (i.e., 10 source nodes, and 50 nodes, respectively). In this experiment, the average value of PDR was 96% for various mobility values (0-10 m/s speed). The small percentage of packet drop, which is 4% is attributed to two different factors: (i) link breakages resulting from the mobility, an outcome also observed in the operation of pure AODV, and, (ii) false positive alarms triggered by the proposed mechanism (see section 4.3.1).

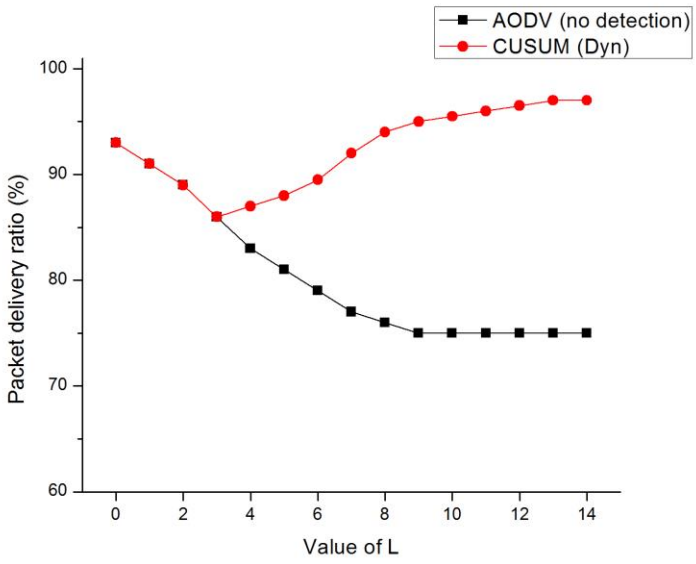


Figure 9: PDR as a function of the value of L with a fixed traffic flow of 10 source nodes

4.3.4 Comparative evaluation with other detection mechanisms

Following the experimental analysis of the proposed detection mechanism and the identification of its optimal parameters, in this section, we provide a comparative evaluation, in which we compare the performance of the proposed detection mechanism to other security mechanisms, designed for the detection of blackhole attacks. This evaluation, in addition to the proposed detection mechanism, also includes results from: the GAODV protocol [17], the PCBHA reputation scheme [16], three detection mechanisms (DCM [13], SVM [14], and ABM [18]), as well as the pure AODV, running in the absence of any security mechanisms. The latter is used as a base, where the imposed overheads by the considered security mechanisms can be compared and studied. GAODV and PCBHA were chosen because they constitute solutions that encompass security features within AODV. Moreover, the three detection mechanisms were selected since they utilize state-of-the-art techniques as well as they provide a broad set of simulation results, allowing for a detailed comparison with the proposed detection mechanism.

In order to facilitate the comparison among the studied solutions, we have selected the following metrics: (i) the PDR (i.e., the percentage of transmitted packets that reach their destination) as a function of the number of malicious nodes; (ii) the observed detection accuracy (i.e., the percentage of attacks that are resolved by the detection mechanism) as a function of the average node's speed; (iii) the imposed control packet overhead (i.e., the increase in the percentage of control packets transmitted by AODV) as a function of the number of packets; and (iv) the computational complexity. The PDR, in the presence of a malicious node(s) performing a blackhole attack(s), may assess the ability of the considered mechanism to detect and isolate malicious nodes, minimizing the effects of the attack(s) in the operation of the

network. The detection accuracy of the underlying mechanism, in the presence of a blackhole attack(s) and under variable nodes' mobility, evaluates the mechanism's ability to resolve attacks in volatile network conditions. The control packet overhead measures the corresponding control packets initiated by the considered mechanisms, determining the induced communication overhead. Finally, the computational complexity quantifies the amount of time taken by each detection mechanism to run; in relation to the input taken, and thus, determines its resource requirements (i.e., processing and energy). The simulation parameter values were selected based on the values used by the related work, in order to generate similar experimental conditions, and thus, provide comparable results.

Figure 10 presents the PDR as a function of the number of malicious nodes, performing a packet dropping attack for AODV (i.e., with no detection), the proposed detection mechanism, ABM, GAODV, and DCM. The experiment was conducted under fixed mobility, traffic flow, and number of nodes (i.e., 10 m/s, 10 source nodes, and 50 nodes, respectively) so that the PDR was only affected by the number of malicious nodes. The objective of this experiment is to assess the scalability of the evaluated detection mechanisms, when more than one malicious nodes are present in the network. SVM and PCBHA are excluded from this experiment, since their authors do not provide such results for their relative performance. In the presence of one malicious node, the PDR of pure AODV (i.e., without any security measure) is about 80%; while for 10 malicious nodes the ratio drops down to 20%, depicting the impact of the attack. The PDR for the proposed detection mechanism (i.e., the dynamic threshold CUSUM) is approximately 98%, regardless of the number of malicious nodes, the density of network nodes, or the volume of generated traffic. These results confirm the assertion that by monitoring the exploitation step of the attack, instead of packet dropping, the blackhole attack is detected, immediately, after its initiation by the malicious node. The employment of the DCM mechanism degrades the network operation and the considered metric as the number of malicious nodes increases from 1 to 10, reaching the lowest value of 80% for the presence of 10 malicious nodes. ABM demonstrates similar performance and as the number of malicious nodes increases from 1 to 10, it reaches the lowest value of 73% for the presence of 10 malicious nodes. On the contrary, with GAODV, in the presence of one malicious node, the PDR drops to 91%, but remains at that value (on average) regardless of the number of malicious nodes. This occurs because in GAODV, source nodes await for an acknowledgment packet from the destination. The duration until the acknowledgment packet is received, enables the malicious node to drop some of the received traffic.

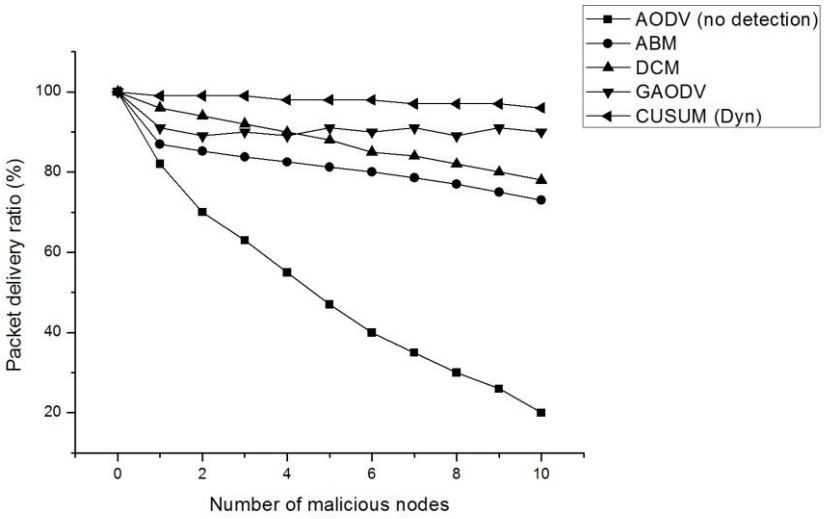


Figure 10: PDR as a function of the number of malicious nodes

Figure 11 presents the detection accuracy of the dynamic threshold CUSUM variant and SVM as a function of the average nodes' speed. The goal of this experiment was to evaluate the ability of the detection mechanisms to identify blackhole attacks, under high network volatility. ABM, GAODV, DCM, and PCBHA are excluded from this study, since their authors do not provide such results for their performance. We conducted 6 iterations of this experiment, in each of which we increased the average node's speed by a factor of 2 m/s, beginning with an initial value of 0 m/s, up to a maximum of 10 m/s. For each iteration, one node was randomly selected to perform a blackhole attack. The experiment was conducted under fixed traffic flow and number of nodes (i.e., 10 source nodes, and 50 nodes, respectively).

The employment of CUSUM results in a detection accuracy that does not change (i.e., remains on average at 98%), regardless of the volatile network conditions (i.e., node movement, density of nodes, or volume of traffic). Again, the dynamic threshold CUSUM maintains a fixed detection accuracy rate under volatile network conditions, due to the ability of adjusting the threshold value and perform "retraining" when the average nodes' speed on the network varies. SVM performed comparatively well, as it exhibited only a mirror drop. Its average detection accuracy ranged from 98% (no mobility), to 95% (10 m/s speed). The performance of SVM is attributed to the use of a dynamically updated normal profile.

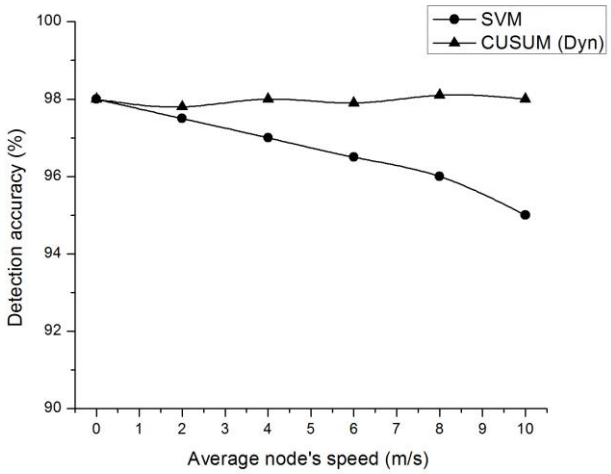
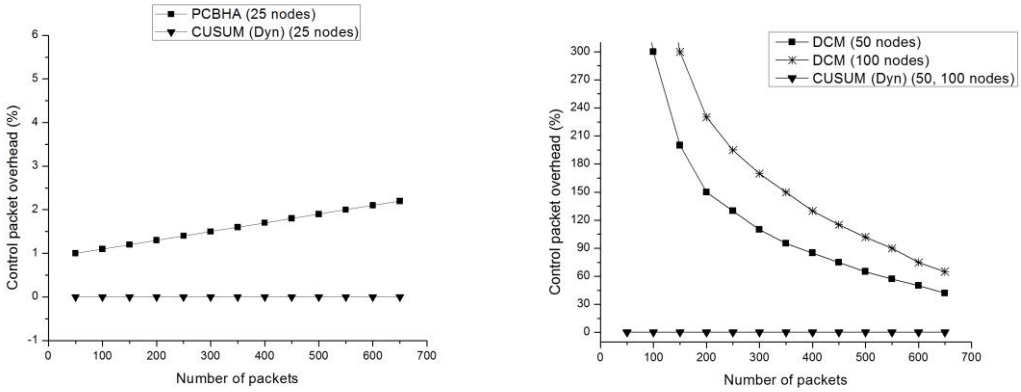


Figure 11: Detection accuracy as a function of the average node's speed

The subsequent experiments depict the percentage of additional control packets, compared to pure AODV, transmitted by the proposed detection mechanism and PCBHA (see Figure 12a), as well as the proposed detection mechanism and DCM (see Figure 12b), as a function of the total number of transmitted packets. These experiments demonstrate the communication overhead (if any) induced by the evaluated detection mechanisms. In both experiments, the average node speed ranged from 0 m/s to 10 m/s, while the number of traffic flows remained fixed (i.e., 10 source nodes). Furthermore, we conducted one iteration of the first experiment (presented in Figure 12a), in which the total number of network nodes was equal to 25, and two iterations for the second experiment (presented in Figure 12b), in which the total number of network nodes was equal to 50 and 100 nodes, respectively. Once again, ABM, SVM, and GAODV are excluded from this study, since their authors do not provide comprehensive results for their performance.

In both experiments, the proposed detection mechanism does not induce any control packet overhead, since each node executes its own instance of CUSUM and no cooperation between instances takes place. The employment of PCBHA in a network of 25 nodes resulted in small control message overhead ranging from 1% to 2% (see Figure 12a). This overhead is attributed to the fact that nodes have to exchange additional information before selecting a route (i.e., reputation scores). DCM, on the other hand, exhibited a control packet overhead, exceeding 300% in both a network of 50 nodes and a network of 100 nodes (see Figure 12b). However, the occurred overhead diminishes as the total number of transmitted packets increases. This is mainly caused by the mechanisms' ability to provide more accurate decisions about the behavior of nodes as the respective number of monitoring packets increases, minimizing in this way the need to transmit "test packets" between monitoring nodes and thus, reducing the associated communication overheads.



Figures 12a and 12b: Control packet overhead as a function of the number of packets

Table 7 outlines the computational complexity of the proposed detection mechanism in comparison to DCM, SVM, ABM, and PCBHA. GAODV is excluded from this comparison, since its authors do not provide any associated computational complexity estimate, nor could we derive one from the provided algorithm. Note that each of the compared mechanisms follows a different approach in detection, and, thus, utilizes different sets of input. Based on the analysis presented in section 4.2, the dynamic threshold CUSUM variant induces linear computational complexity, for an input of n_1 route table entries (please note that the notation n_1 is identical to K which is used throughout the paper to denote the number of route table entries). On the other hand, all of the other evaluated mechanisms are characterized by higher computational complexity. In particular, the computational complexity of DCM is quadratic for an input of n_2 captured data packets, since, at the worst case, for each captured packet, it must evaluate n_2 entries of the "estimation table", broadcast to n_2 neighboring nodes, generate a table of n_2 replies, and traverse it to compute the voting result. ABM has a quadratic computational complexity, for an input of n_3 route requests, since the detection algorithm traverses a "RQT_Table" of size n_3, twice per route request. Next, PCBHA demonstrates a quadratic computational complexity as well (for an input of n_4 route requests), since, for each route request, the detection algorithm must traverse 2 different n_4 sized tables (i.e., the fidelity and response tables). Finally, SVM, according to its authors, has a cubic computational complexity, caused mainly by its re-training mechanism. SVM collects a variety of features, such as the number of neighboring nodes, the delay of Hello messages, and the difference of sequence numbers between Hello messages. Subsequently, a classifier converts the collected features into n_5 support vectors, which are then used as input by the SVM detection algorithm.

Table 7: Computational complexity of the evaluated detection mechanisms

Detection mechanism	**CUSUM (Dyn)**	**DCM**	**ABM**	**PCBHA**	**SVM**
Input	n_1 routing table entries	n_2 data packets	n_3 route requests	n_4 route requests	n_5 support vectors
Computational complexity	Linear $O(n_1)$	Quadratic $O(n_2^2)$	Quadratic $O(n_3^2)$	Quadratic $O(n_4^2)$	Cubic $O(n_5^3)$

5 Conclusions

This paper provided a comprehensive analysis of the blackhole attack, identified a new critical attack parameter (i.e., blackhole intensity), and evaluated the impact of that parameter to the performance of the attack, through an extensive set of simulations. Based on the results of the simulations, we identified a quantitative relation between SQNs and blackhole attacks. This outcome led to the proposal of a novel detection mechanism, which utilizes a dynamic threshold cumulative sum (CUSUM) test to detect abrupt changes in the normal behavior of SQNs. A key advantage of the proposed mechanism is its ability to accurately detect attacks with a minimal rate of false positives, even if the malicious node selectively drops packets. Furthermore, the proposed mechanism passively monitors AODV protocol operations, thus inducing no additional latency to the routing protocols' operation. The performance of the proposed detection mechanism was comparatively evaluated through an extensive set of simulations. The simulation results demonstrated that the proposed mechanism resolves attacks with high detection accuracy for blackhole intensity values above four. It is also resilient to conditions of high network volatility, and it alleviates the need for communication overheads, often exhibited by other detection mechanisms that have been proposed in the literature, while inducing the least amount of computational overhead.

Acknowledgments

This research has been partially funded by the ReCRED project (Horizon H2020 Framework Programme of the European Union under GA number 653417).

References

[1] C. Panos, C. Xenakis, P. Kotzias and I. Stavrakakis, "A specification-based intrusion detection engine for infrastructure-less networks," Computer Communications, 54, 67-83, 2014.

[2] C. Perkins, E. Belding-Royer, and S. Das, "Ad hoc On-Demand Distance Vector (AODV) Routing," IETF RFC 3561, Jul. 2003.

[3] J. Ott, D. Kutscher, and C. Dwertmann. Integrating DTN and MANET Routing. In CHANTS '06: in proceedings of the 2006 SIGCOMM Workshop on Challenged Networks, pages 221–228, New York, NY, USA, 2006.

[4] J. Lakkakorpi, M. Pitkanen, and J. Ott, "Adaptive routing in mobile opportunistic networks," in proceedings of the ACM MSWIM'10, (Bodrum, Turkey), pp. 101–109, October 2010.

[5] A.A. Pirzada, M. Portmann, J. Idulska, “Evaluation of Multi-Radio Extensions to AODV for Wireless Mesh Networks”, in proceedings of the international workshop on Mobility management and wireless access ACM MobiWac 2006.

[6] C. Gomez, P. Salvatella, O. Alonso, J. Paradells, “Adapting AODV for IEEE 802.15.4 mesh sensor networks: theoretical discussion and performance evaluation in a real environment,” in proceedings of the 2006 International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM’06), 2006, pp. 159–170.

[7] B. Wu, J. Chen, J. Wu, and M. Cardei, “A Survey of Attacks and Countermeasures in Mobile Ad Hoc Networks” in “Wireless Network Security”, Y. Xiao, X. Shen, and D. -Z. Du, Springer, Network Theory and Applications, Vol. 17, 2006.

[8] C. Xenakis, C. Panos, I. Stavrakakis, “A comparative evaluation of intrusion detection architectures for mobile ad hoc networks,” Computers & Security, Volume 30, Issue 1, January 2011.

[9] B. Kannhavong, et al. "A survey of routing attacks in mobile ad hoc networks." Wireless communications, IEEE Vol. 14: pp: 85-91, 2007.

[10] A. Bala, B. Munish, and S. Jagpreet, "Performance analysis of MANET under blackhole attack." Networks and Communications, 2009. NETCOM'09. First International Conference on, pp. 141-145. IEEE, 2009.

[11] S. Sharma, R. Gupta, "Simulation study of blackhole attack in the mobile ad hoc networks." Journal of Engineering Science and Technology4, Vol. 2, pp 243-250, 2009.

[12] E. Barkhodia, S. Parulpreet, and G. K. Walia. "Performance analysis of AODV using HTTP traffic under Black Hole Attack in MANET."Comput. Sci. Eng. Int. J.(CSEIJ) 2, Vol. 3, 2012.

[13] C.W. Yu, T.K. Wu, R. H., Cheng, and S. C. Chang, “A Distributed and Cooperative Black Hole Node Detection and Elimination Mechanism for Ad Hoc Networks”, PAKDD 2007 Workshops, pp. 538–549, 2007.

[14] J.F.C., Joseph, Bu-Sung Lee, A., Das, Boon-Chong Seet, “Cross-Layer Detection of Sinking Behavior in Wireless Ad Hoc Networks Using SVM and FDA,” Dependable and Secure Computing, IEEE Transactions on, vol.8, no.2, pp.233-245, March-April 2011.

[15] Payal N. Raj, Prashant B. Swadas: Dpraodv: A Dynamic Learning System Against Blackhole Attack in AODV Based Manet, CoRR abs/0909.2371, 2009.

[16] LathaTamilselvan, Dr, V Sankaranarayanan, “Prevention of Co-operative Black Hole Attack in MANET”, Journal of Networks, vol. 3, No. 5, pp. 13-20, May 2008.

[17] Sanjay K. Dhurandher, Isaac Woungang, RaveenaMathur, PrashantKhurana, “GAODV: A Modified AODV against single and collaborative Black Hole attacks in MANETs”, 27th International Conference on Advanced Information Networking and Application Workshops, pp. 357-362, March 2013.

[18] Ming-Yang Su, “Prevention of selective blackhole attacks on mobile ad hoc networks through intrusion detection systems”, Computer Communications, vol. 34, pp. 107-117, January 2011.

[19] M. Basseville and I.V. Nikiforov, Detection of Abrupt Changes: Theory and Application, Prentice Hall, 1993.

[20] B.E Brodsky and B.S. Darkhovsky, Nonparametric Methods in Change-Point Problems, Kluwer Academic Publishers, 1993.

[21] Patrick P. C. Lee, Tian Bu, Thomas Y. C. Woo: On the detection of signaling DoS attacks on 3G/WiMax wireless networks. Computer Networks (CN) 53(15):2601-2616 (2009).

[22] ARM Architecture Reference Manual, Thumb-2 Supplement. Available online at: http://read.pudn.com/downloads159/doc/709030/Thumb-2SupplementReferenceManual.pdf

[23] Intel 64 and IA-32 Architecture Software Developer’s Manual. Available online at: http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-software-developer-instruction-set-reference-manual-325383.pdf

[24] Texas Instruments M4C123x Microcontroller datasheet based on ARM's Cortex-M4F reference design. Available online at: http://www.ti.com/lit/ds/symlink/tm4c123gh6pz.pdf

[25] Veeraraghavan, P., Vikram L., "Trust in mobile ad hoc networks." Telecommunications and Malaysia International Conference on Communications, 2007. ICT-MICC 2007. IEEE International Conference on. IEEE, 2007.
[26] Tao, P., Leckie, C., and Ramamohanarao, K., "Proactively detecting distributed denial of service attacks using source IP address monitoring." International Conference on Research in Networking. Springer Berlin Heidelberg, 2004.
[27] Siris, V., and Papagalou. F., "Application of anomaly detection algorithms for detecting SYN flooding attacks." Computer communications 29.9 (2006): 1433-1442.
[28] P. Gupta, R. K Tuteja, "Design Strategies for AODV Implementation in Linux" in International Journal of Advanced Computer Science and Applications(IJACSA), vol. 1, issue. 6 2010
[29] Ian D. Chakeres, Elizabeth M. Belding-Royer, AODV Routing Protocol Implementation Design, Proceeding of IEEE 24th International Conference on Distributed Computing Systems Workshops, pp: 698-703, March 2004.