S-Octopus: A Novel Scalable Secure Position-Based Routing Protocol for MANETs

Liana K. Qabajeh
Faculty of Information Technology and Computer Engineering (CITCE), Palestine Polytechnic University, Hebron, Palestine
Email: liana_tamimi@ppu.edu (L.K.Q.)

Abstract **—Mobile Ad-Hoc Networks (MANET) are self-organized wireless networks that are becoming progressively popular. Determining an efficient route leading from a source to a specific destination in these networks is an essential issue since nodes are continuously moving. Furthermore, finding a secure route is a difficult area to deal with since adversaries might insert themselves into these routes unless a strict secure routing procedure is implemented. In this paper, a novel scalable secure routing protocol called S-Octopus has been proposed. Via dividing the network area into sectors and utilizing restricted directional flooding, our protocol intents to achieve improved scalability. Moreover, S-Octopus seeks to enhance robustness against the single point of failure and compromise by introducing several Sector Certificate Authority servers. Together with S-Octopus a location service and a misbehavior detection system have been proposed. Using GloMoSim simulator, S-Octopus security and performance have been evaluated and compared with the basic Authenticated Routing for Ad-Hoc Networks (ARAN) as well as Zone-based Authenticated Routing for Ad-Hoc Networks (ARANz). Simulation results assure that S-Octopus is able to effectively initiate and maintain secure routes in MANETs. Results also confirm that S-Octopus has significantly mitigated the scalability problem by achieving the maximum packet delivery fraction and the minimum network and routing loads within fairly large networks with high-mobility nodes and large malicious node percentage. Thus, S-Octopus is a good choice for MANET established among students on a campus or peers at a conference, where keys and certificates might be previously deployed.**

Keywords**—mobile, scalable, secure, position-based, routing, ad-hoc networks, MANETs, managed-open networks**

I. Introduction

A Mobile Ad-Hoc Network (MANET) is a multi-hop self-regulated wireless network. Every node in a MANET participates in forwarding packets and is moving rapidly in most cases [1, 2]. MANETs may be established dynamically on demand, since they do not require pre-established infrastructure such as routers. Accordingly, they are implemented in various areas, including community and emergency networks [3–5]. Guaranteeing efficient and secure routing in MANETs is a crucial issue. Upon implementing such networks, it is essential to reduce transmission overhead since wireless links usually have low-bandwidth and nodes rely on batteries and usually have limited processing capacity and memory [6, 7]. Moreover, MANETs are susceptible to attacks via various techniques as modification, impersonation, and fabrication [8].

In managed-open environment [9, 10], such as that established by students on a campus, or employees in a factory, using already established infrastructure is probable [11]. This means that there is an opportunity to give a starting point for assuring security in such networks by pre-deploying public keys, session keys, or certificates. However, depending on a single centralized server in MANETs is impractical as this server may move rapidly resulting in a difficulty for other nodes to connect to it. Additionally, the server could be the operation bottleneck as it could be just a normal mobile node with limited capabilities. To address such a problem, the certificate authority must be spread among multiple servers. Finally, it has been noticed that position-based routing approaches have been widely introduced in MANET, especially due to the necessity of scalable and energy-efficient protocols, along with the existence of low-cost and low-power positioning devices [9, 10].

The aforementioned discussion encourages us to propose S-Octopus to provide a scalable and secure MANET routing protocol. S-Octopus aims to improve performance and distribute routing load by dealing with the network area as sectors. Moreover, S-Octopus tries to attain satisfiable level of robustness and security, along with avoiding single point of failure and attack problems via distributing trust among multiple sector certificate authorities. Moreover, S-Octopus seeks to show high level of scalability and performance through using restricted flooding. So, along with S-Octopus a location service is proposed. Finally, S-Octopus utilizes a misbehavior detection system to eradicate malicious nodes.

Our methodology in this research is to try to answer the following research questions:

(1) Will dealing with the network as sectors and introducing numerous certificate authorities help S-Octopus to achieve satisfiable performance and scalability?
(2) Will identifying and isolating the malicious nodes in S-Octopus help in achieving high level of performance and security?

Manuscript received September 5, 2022; revised November 23, 2022, accepted January 23, 2023.

Hence, we can set out and try to prove our research hypotheses:

(1) S-Octopus is able to achieve high-level of performance and scalability via dividing the network into sectors and introducing several certificate authorities.

(2) Implementing S-Octopus along with the proposed misbehavior detection system improve the performance and security of the network.

This paper is an extension of our work in [11]. A performance and security analysis of S-Octopus, Authenticated Routing for Ad-Hoc Networks (ARAN) [12] and Zone-based Authenticated Routing for Ad-Hoc Networks (ARANz) [3] protocols have been presented in [11]. This paper, on the other hand, presents a detailed simulated network performance and security evaluation and comparison among S-Octopus, ARAN and ARANz protocols. Using GloMoSim simulator, the effect of five important parameters of MANETs have been tested. These parameters are node mobility speed, network size, nodes density, local communication percentage, and malicious node percentage.

The simulation results assure that S-Octopus discovers secure routes effectively within relatively large networks, high-mobility nodes and large number of malicious nodes. Also, S-Octopus assures scalability by achieving the maximum packet delivery fraction and the minimum network and packet routing load in most scenarios.

The rest of the paper is structured as follows. Section II discusses the related works on MANET routing protocols. Section III discusses our newly proposed routing protocol. Sections IV presents security analysis along with a simulated comparison between ARAN, ARANz and S-Octopus protocols. Our findings are discussed and our work is concluded in Section V and Section VI respectively. Finally, our future directions are presented in Section VII.

II. Background and Related Works

Various routing protocols have been suggested for MANETs. Generally, they are classified into topology-based and position-based protocols. *Topology-based* protocols utilize information about links currently in the network to forward packets. Topology-based protocols are further classified into proactive, reactive, and hybrid protocols. *Proactive* protocols are less appropriate for MANETs since periodic broadcast of control packets consumes network power regardless of the existence of network activity [13–15]. In contrast, ***reactive*** protocols start a route discovery only upon the need to send data packets. Many reactive routing protocols were suggested including Ad-Hoc On-demand Distance Vector (AODV) [5] protocol. These protocols do not involve periodic routing packets; however, they may have increased control overhead in heavy-load and high-mobility cases. Scalability is considered as another disadvantage since they use blind broadcasts to discover routes [15]. *Hybrid* protocols, like Zone Routing Protocol (ZRP) [14], aim to combine advantages of both proactive and reactive methods [16]. Topology-based approaches are appropriate for networks containing few hundreds of nodes [17]. In addition, the abovementioned protocols trust all participating nodes, which may allow security vulnerabilities and attacks [18, 19].

After that, many works have considered routing protocols security. Some of them, including [20–23], have discussed a deep analysis of MANETs security issues and presented the proposed defeating techniques against existing attacks. Some other researchers conducted a security assessment of some existing MANETs secure routing protocols. Authors in [24], for example, examined the performance of AODV protocol under several security attacks. They found that conducting different attacks reduces throughput and packet delivery ratio. Furthermore, authors in [25], studied the performance and security of AODV routing protocol and Secure Ad-Hoc On-demand Distance Vector routing protocol [26] bearing in mind various attacks such as blackhole and replay.

Other researchers suggested new security procedures to circumvent particular attacks. In [27, 28], new flooding attack prevention protocols are presented. In [29] a secured scheme has been suggested for networks implementing reactive protocols as AODV. Each node computes the trust in view of both the direct and the collected information from its neighboring nodes. Integrating the proposed procedure at every node enhances the throughput and reduces the overhead.

In [30], authors proposed a quantitative trust system for Ad-Hoc networks that are integrated with Internet of Things (IoT). The proposed system combines both direct and indirect trust to calculate a node’s trust value. Additionally, only trusted nodes are chosen in the route from source to destination. Detailed surveys of recent work conducted on security solutions for MANETs are presented in [8, 20–22, 31, 32].

One protocol of concern is the *Authenticated Routing for Ad-Hoc Networks (ARAN)* [12] protocol; since it assures route discovery, setup and maintenance authentication along with message integrity and non-repudiation. Moreover, ARAN prevents a number of attacks including modification, impersonation and fabrication. ARAN is a secure extension of AODV; hence, no periodic routing packets are required. ARAN aims to guard against exploits from misbehaving nodes in managed-open environments where prior security coordination is possible. It depends on a trusted Certificate Authority (CA) server whose public key is distributed to all other nodes. All nodes are supposed to request a certificate from this CA.

ARAN starts route discovery via broadcasting a Route Discovery Packet (RDP) by the source, which is replied to via a unicast REPly (REP) packet from the destination node back towards the source node. Routing packets are authenticated at every hop from source to destination and from destination to source. ARAN requires that participating nodes keep one routing entry per source-destination active pairs. This definitely results in higher cost compared to per-destination entries in non-secure protocols. Along with its scalability problem with the number of nodes, it introduces packet overhead and latency in route discovery due to signing packets.

Furthermore, it depends on a centralized trust and faces the single point of failure and the compromised server.

After that, *position-based* protocols show higher level of scalability and robustness against common topological changes [33, 34]. Position-based protocols utilize information about nodes geographical positions to take routing decisions, in a try to improve performance and efficiency. This category requires all nodes to attain their own geographical positions via Global Positioning System (GPS) and the destination geographical position using location service. Position-based protocols are classified into greedy, restricted directional flooding and hierarchical protocols.

In *Greedy forwarding* approaches, such as Greedy Perimeter Stateless Routing (GPSR) [35], intermediate nodes forward packets from source to destination by selecting the neighboring node that has the minimum cost towards the destination as their successor. This procedure continues until the data reaches the destination. Hence, nodes periodically broadcast small beacons to declare their positions and assist other nodes in keeping a one-hop neighbor table. These protocols are considered scalable since they do not necessitate route discovery and maintenance [36]. However, periodic beaconing consumes the nodes energy and results in network congestion [17, 33]. Additionally, Greedy forwarding generally are not assured to find the ideal route [36]. For instance, GPSR works well in dense networks, but not in sparse ones [35].

Location-Aided Routing (LAR) [37] is a *restricted directional flooding* protocol in which the packet is sent by the source to all single hop neighbors towards the destination. Each node, upon getting a route request, compares the distance from itself to the destination, with the distance from its preceding node to the destination. If the node is closer to the destination, the packet is retransmitted; else, it is dropped. Each intermediate node includes its IP address in the packet header resulting in increased packet size.

TERMINODES [38] is an example of ***hierarchical*** protocols that utilize a two-level hierarchy. If the destination is close to the sender, a packet is routed using a proactive distance vector. Instead, greedy forwarding is used for distant routing. All the above-mentioned position-based protocols are susceptible to numerous security attacks [19]. Moreover, many of them have little probability of finding the shortest path.

After that, several secure position-based routing protocols have been suggested including *Zone-based Authenticated Routing for Ad-Hoc Networks (ARANz)* [3]. ARANz adopted the ARAN authentication methods aiming to increase security and prevent most attacks against MANET routing protocols. Nevertheless, ARANz proposed a hierarchal routing procedure, aiming to improve performance and share out load via handling the area as square-shaped zones. Additionally, it tried to achieve robustness and solve the single point of attack and failure problems via choosing numerous Local Certificate Authority (LCA) servers. Each zone has several LCAs collaborating together to issue certificates to the nodes within that zone and working as backup nodes for each other. ARANz network structure is illustrated in Fig. 1, supposing that the network area is divided into 3×3 zones. Using restricted directional flooding helped ARANz in exhibiting increased performance and robustness against fast topology changes. Whenever a node needs to send data, the source gets the destination position via communicating its zone LCAs. After that, the route request packet is sent via restricted directional flooding to mitigate overhead and save network bandwidth, compared to original ARAN protocol. Hence, each node should inform its local LCAs about its new position if it moved. The first phase of ARANz is network setup phase, which consists of issuing certificates to trusted nodes, dividing area into zones and choosing initial LCAs. Network maintenance phase aims to ensure maintaining the network structure considering some issues like updating nodes certificates, synchronizing LCAs, and updating nodes positions.

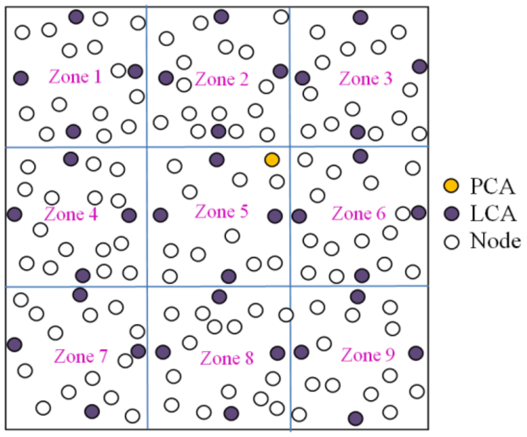


Figure 1. ARANz network structure.

Location service phase facilitates obtaining the destination position by the source node via communicating LCAs in its zone. After attaining the destination position, ARANz initiates the route instantiation and maintenance phase. A Route Discovery Packet (RDP) is sent by the source using restricted directional flooding to the destination. A Route REPly (RREP) packet is unicast by the destination upon receiving the first RDP. RREP packet is sent back along the reverse path to setup the route. As a following step, the source node starts sending the data to the anticipated destination node. To maintain the chosen routes, nodes in ARANz use Error (ERR) packets to notify source about broken links within active routes.

All packets are signed by the sources and intermediate nodes using their private keys and nodes certificates are attached to the packets. Moreover, all intermediate nodes and the destination nodes, validate the preceding node signature using its public key that is extracted from the appended certificate. Thus, it is assured that packets sent during the route instantiation are end-to-end authenticated. Consequently, data packets exchanged among nodes are unsigned and do not include certificates.

Dividing the network into multiple square-shaped zones and assigning four LCAs on the boundaries of each zone, make communication between LCAs of adjacent zones faster and easier. However, this results in increased control overhead and latency in performing the operations that require communication among LCAs within a particular zone such as updating nodes certificates, updating nodes positions, obtaining destination position, LCAs synchronization, and announcing malicious or compromised nodes.

To summarize, many topology-based routing protocols have scalability problems and security vulnerabilities. In spite of proposing some improvements on security features such as in ARAN, the sole centralized node trust has introduced other security issues and affected scalability. Finally, restricted directional flooding assures better performance compared to topology-based and other position-based protocols. ARANz is an instance of restricted directional flooding, however, introducing multiple local servers and maintaining network structure result in extra control overhead.

III. S-Octopus Protocol

In this section, our newly proposed routing model S-Octopus is presented. S-Octopus, just like ARAN and ARANz, uses cryptographic certificates to avoid most attacks. S-Octopus aims to enhance performance and distribute load by dealing with the area as sectors. Moreover, it looks forward to improve robustness and security by distributing trust among multiple Sector Certificate Authority (SCA) servers. S-Octopus tries to select the SCAs to be as close as possible to the network center to reduce the overall overhead and latency resulted from SCAs communications. Additionally, adjacent SCAs act as a team to issue nodes certificates. S-Octopus also proposes a misbehavior detection scheme to discover the malicious and compromised nodes. S-Octopus uses Restricted directional Flooding (ResF) to enhance scalability, performance, and robustness. Whenever there is data to be sent between nodes, the source should get the destination's position via contacting its sector SCA. After that, the route request packet is forwarded using ResF to reduce overhead and save bandwidth. Each participant node should keep the SCA of its sector aware of its position; as the SCAs also work as Position Servers.

S-Octopus consists of six stages; network setup, network maintenance, location service, route instantiation and maintenance, data transmission, and lastly misbehavior detection system. During *Network setup*, trusted nodes are certified, network area is divided into sectors and SCAs are elected. *Network maintenance* stage maintains the network structure considering updating nodes certificates, SCAs synchronization, and nodes movements. During *Location service*, the source communicates its sector SCA to obtain the destination position. This SCA may communicate other SCAs if needed. After that, the *route* instantiation *and maintenance* stage is started. So, the source issues a route discovery packet using ResF towards the destination. When receiving the first route discovery packet, destination sends a route reply packet back along the Reverse Path (RevP) to the source to setup the selected route. After that, the source starts *sending the data* towards the destination. Nodes in S-Octopus use error packets to maintain active routes and report broken links. A *misbehavior detection system* is used to detect misbehaving nodes in the network. The details of different stages of our protocol are presented in the following subsections. Table I shows variables and notations used with S-Octopus. Table II summarizes the keys involved in S-Octopus. Whereas, Table III summarizes different certificates used with it. Table IV summarizes the different techniques used to forward packets, while Table V shows the strategies for sending packets during different stages of S-Octopus.

TABLE I. Variables and Notations for S-Octopus

Notation	Description	Notation	Description
N	Nodes number	S	Sectors number
L	Network area Length	W	Network area Width
PCA	Primary Certificate Authority	SCA_s	Sector Certificate Authority of sector s
S_s	Sector number s	CK	Common Key
K_{NET-}	Network private key	K_{NET+}	Network public key
K_{Nx-}	Node × private key	K_{Ss-}	Sector s private key
K_{Nx+}	Node × public key	K_{Ss+}	Sector s public key
$Cert_{Nx}$	Node × Certificate	$Cert_{Ss}$	Sector s SCA Certificate
IP_x	Node × IP address	P_x	Node x Position
N_x	Nonce issued by node x	Pr_x	Probability of node x to be elected as SCA
S_x	Node × Speed	B_x	Node x Battery remaining life time
C_x	Node × CPU power	M_x	Node x Memory
$Role_x$	Node × current role	AT	Authentication Table
t	Timestamp of certificate creation	e	Certificate expire time
Adj_{Ss}	Identity, position and public key of Adjacent SCAs (predecessor and successor) of sector s	Adj_{Ns}	IP address and certificate expiration date for Nodes in the predecessor sector) of sector s
SR	Source Route that a packet will go through	Dist	Distance between an intermediate node and the destination node
d_{mov}	Pre-defined distance that a node is allowed to move from its most recent position before it must send its new position to its SCA	d_{cen}	Pre-defined distance that a SCA is allowed to move from the network center before it should initiate a new SCA election
NetF	Network Flooding	SecF	Sector Flooding
ResF	Restricted directional Flooding	SrcR	Source Routing
RevP	Reverse Path	RlyD	Data packet Relaying
$TVMod_{xy}$	Modification attack node x trust value regarding node y	ThMod	Modification threshold
$TVDrop_{xy}$	Dropping attack node x trust value regarding node y	ThDrop	Dropping threshold

$TVFab_{xy}$	Fabrication attack node x trust value regarding node y	ThFab	Fabrication threshold
$UnModP_{xy}$	Number of unmodified control packets sent by node y and received by node x	$ModP_{xy}$	Number of modified control packets sent by node y and received by node x
$UnDropP_{xy}$	Number of delivered data packets by node y that it received from node x	$DropP_{xy}$	Number of dropped data packets by node y that it received from node x

TABLE II. DIFFERENT KEYS USED WITH S-OCTOPUS

Key	**Owned by**	**Used for**
Common key (CK)	All nodes.	Encrypting and decrypting messages sent by non-PCA nodes during setting up the network.
Node private/public key pairs (K_{Nn-}/K_{Nn+})	Each particular node n.	• Encrypting and decrypting control packets sent by node n after setting up the network.
Network private/public key pair (K_{NET-}/ K_{NET+})	• Public key is owned by all nodes. • Private key is owned by PCA and all SCAs.	• Encrypting and decrypting packets sent by PCA during setting up the network. • Encrypting and decrypting certificates of the nodes.
Sector private/public key pairs (K_{Ss-}/ K_{Ss+})	• Public key of a particular sector is owned by nodes residing in that sector. • Private key is owned by SCAs of that sector.	Encrypting and decrypting a particular sector SCA certificate.

TABLE III. DIFFERENT TYPES OF CERTIFICATES USED WITH S-OCTOPUS

Certificate	**Issued by**	**Issued to**	**Used for**
Node certificate ($Cert_{Nn}$)	SCA of the sector where node n resides.	Each trusted node n	Nodes authentication during different stages of S-Octopus.
Sector SCAs certificate ($Cert_{Ss}$)	SCA of the sector where the SCA resides.	SCAs	SCAs verification during different stages of S-Octopus.

TABLE IV. PACKETS FORWARDING TECHNIQUES USED WITH S-OCTOPUS

Forwarding technique	**Notation**	**Description**	**General packet structure**
Network Flooding	$S \xrightarrow{NetF} ALL$	•Flooding packet to all nodes currently in the network. •Any node continues broadcasting the packet upon receiving it.	[Pid, …]K_{Ss-}, $Cert_{Ss}$
Sector Flooding	$S \xrightarrow{SecF} ALL_s$	•Flooding packet to all nodes existing currently in sector s. •Only nodes residing currently in sector s process and continue broadcasting the packet.	[Pid, s, …]K_{Ss-}, $Cert_{Ss}$
Restricted directional Flooding	$S \xrightarrow{ResF} D$	•Sending packet using restricted directional flooding. •Intermediate node continues broadcasting the packet if it is closer to D than its predecessor.	[Pid, IP_D, …]K_{Ss-}, $Cert_{Ss}$
Source Routing	$S \xrightarrow{SrcR} D$	•Sending packet using source routing. •Every node along the route forwards packet to its successor node in SR.	[Pid, SR, …]K_{Ss-}, $Cert_{Ss}$
SCA flooding	$S \xrightarrow{SrcR} D$ or $S \xrightarrow{ResF} D$	•Flooding packet to all SCAs in the network. •Each SCA upon receiving a packet from predecessor SCA sends it to successor SCA. If the successor SCA is reachable in one-hop the packet is sent using source routing, else restricted directional flooding is used.	[Pid, SR, …]K_{Ss-}, $Cert_{Ss}$ or [Pid, IP_D, …]K_{Ss-}, $Cert_{Ss}$
Reverse path	$D \xrightarrow{RevP} S$	•Sending packet through reverse path. •Intermediate node sends the reply packet to the predecessor from which it received the request.	[Pid, IP_S, IP_I, …]K_{ND-}, $Cert_{ND}$
Data packet relaying	$S \xrightarrow{RlyD} D$	•Relaying data packet to its destination. •Intermediate node relays data packets without modification to its next hop in the selected route during the route instantiation.	[DATA, IP_S, IP_D, …]

TABLE V. DIFFERENT S-OCTOPUS PACKETS SENDING STRATEGIES

Stage	**Packets**		**Used strategy**
	From	**To**	
Network setup	PCA	Non-PCA	First packet is sent using NetF. After that, SrcR is used since PCA, at this stage, is acquainted with the positions of all other nodes.
	Non-PCA	PCA	RevP towards PCA.
Network maintenance and location service	Regular	SCA of same sector	ResF.
	SCA	Successor SCA	SrcR containing only the destination IP address (one-hop unicast) if destination is within the source transmission range. Otherwise, ResF is used.
	SCA	Node in same sector	SrcR if the destination is a single node. Otherwise, CluF is used.
Route instantiation and maintenance	Source	Destination	CluF.
	Destination/ intermediate	Source	RevP towards the source.
Data transmission	Source	Destination	RlyD.
Misbehavior detection system	Regular	SCA of same sector	A packet indicating the misbehavior of a node is sent using ResF.
	SCA	ALL	A packet informing that a particular node is compromised is sent using NetF.

A. Network Setup

S-Octopus assumes a managed-open environment containing N willing nodes that are distributed randomly in L×W Km^2 network and know their positions. This area is divided into S sector-shape regions. A specific node is chosen to perform the network setup, divide the network into sectors and select the preliminary SCAs. This node is denoted as the Primary Certificate Authority (PCA) server and owns the network key private part (K_{NET-}). All eligible nodes have a private/public key pair, the network key public part (K_{NET+}) and a Common Key (CK) which is used to encrypt and decrypt packets originated from non-PCA nodes during setting up the network. Keys in managed-open environments are generated in advance and exchanged over an existing relationship between all willing nodes and PCA. For comfort of presentation, Table VI provides the packets sent during the network setup stage.

TABLE VI. PACKETS SENT DURING THE NETWORK SETUP STAGE OF S-OCTOPUS

Packet id	Stand for	Description	From	To
NetSet	Network Setup	• Sent using NetF to notify other nodes of initiating the network setup stage. • Signed using K_{NET-} so that nodes can ensure that the PCA is truly the packet source.	PCA	All non-PCA
NodeInfo	Node Information	• Contains information about the source node such as position, speed, battery remaining life time, CPU power and memory. • Encrypted and decrypted using CK to ensure that this packet is forwarded by authorized nodes only. • Sent through the RevP of the NetSet packet to PCA.	All non-PCA	PCA
NodeRole	Node Role	• A particular packet is unicast to each participant node using SrcR, containing the initial role (SCA or regular node) that this node will play.	PCA	All non-PCA

The network setup is initiated by the PCA via broadcasting a packet notifying different nodes about launching the Network Setup (NetSet) stage. The NetSet packet is sent to all nodes existing currently in the network using Network Flooding (NetF) technique. Supposing that node p has been programmed to be the PCA. It broadcasts the following NetSet packet:

$$\text{PCA} \xrightarrow{NetF} \text{ALL: [NetSet, } IP_p\text{] } K_{NET-}$$

The NetSet packet contains the packet identifier (NetSet) and the source IP address (IP_p). This packet is signed using K_{NET-} to enable nodes to ensure that the PCA is really the sending node. Each node upon receiving the first NetSet packet records the IP address of its previous node, continues broadcasting the packet and replies with a Node Information (NodeInfo) packet to the PCA. NodeInfo packet contains the node IP address (IP_x), along with the required information to select the SCAs such as node position (P_x), speed (S_x), battery remaining life time (B_x), Central Processing Unit (CPU) power (C_x) and memory (M_x). The NodeInfo packets are encrypted using the CK. For example, node *x* sends this packet to PCA (node *p*), assuming that node *y* is the intermediate node from which node *x* received the first NetSet packet:

$$x \xrightarrow{RevP} \text{PCA: (NodeInfo, } IP_x, IP_p, IP_y,$$
$$[P_x, S_x, B_x, C_x, M_x]\ K_{Nx-})\ CK$$

Each node upon receiving a NodeInfo packet tries to decrypt it using CK to ensure that previous node is trusted; otherwise the packet is dropped. After encrypting the NodeInfo packet, it is forwarded through the RevP towards the PCA. The node information contained in the NodeInfo packet is signed using node private key (K_{Nx-}) so that the PCA can asure that the sending node of the packet is really the node claiming that, and to assure the node privacy by ensuring that PCA is the only node that is able to read this private information.

After receiving the NodeInfo packets from all trusted nodes currently in the network, PCA divides the network area into eight sectors (number of octopus legs) and assigns a SCA for each sector. The network structure is shown in Fig. 2. Whereas Fig. 3 shows the arrangement of SCAs as an anti-clockwise ring, which will be used later during the network maintenance, location service, and misbehavior detection system stages.

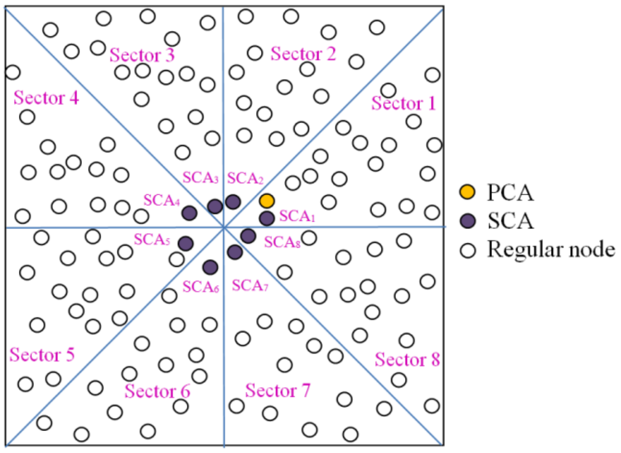


Figure 2. S-Octopus network structure by the end of network setup

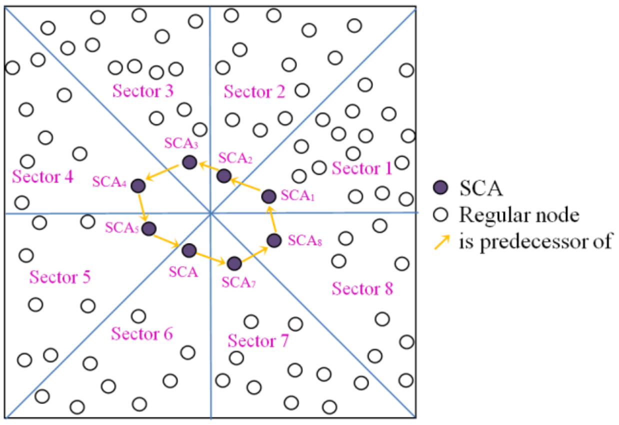


Figure 3. Arrangement of SCAs as an anti-clockwise ring.

This arrangement helps in enhancing S-Octopus robustness and security by avoiding single point of attack and failure. For example, each SCA works as a backup of important information stored in its predecessor in the ring. Hence, each SCA sends all network structure maintenance, certificate update and position update packets to its successor SCA. Moreover, each SCA will not be able to issue certificates to nodes before communicating its two adjacent SCAs and receiving permissions from them.Upon electing SCAs, every node x within a sector S_s is given a weight representing its probability (Pr_x) to be elected as the SCA of that sector. Important points in selecting SCAs are the distance between the node and the network area center point, node speed and battery remaining life time. Selecting a SCA that is close to the center point of the network area and moving slowly raises the probability that these SCAs will be direct neighbors and the communication among them is conducted using single hop. This in turn helps in protecting important packets and reduces overhead. Picking low-speed moving SCA increases the probability that the SCA will stay in the sector for longer time, which in turn reduces the probability of electing a new SCA within a short period of time. Additionally, selecting a node with high battery remaining life time reduces the likelihood of running out its battery energy, i.e.; reduces the probability of electing a new SCA and transferring the important and secure information it possesses. Another two factors to be considered upon electing SCAs are the CPU processing power and nodes memory. Having high CPU processing power and enough memory highly affects the performance of the network since these SCAs may be the operation bottleneck.

After electing SCAs, the PCA unicasts a Node Role packet (NodeRole) to each participating node. Source routing (SrcR) is used for sending these packets since the PCA is aware of all nodes positions. These packets inform nodes of their roles in the network (SCA or regular node). Thus, PCA (node p) sends a unicast NodeRole message to every regular node (non-SCA) x. This message contains the packet identifier (NodeRole), the IP address of the source node (IP_p), the source route (SR) that the packet will pass through, the initial role that the node will play ($Role_x$), the node certificate ($Cert_{Nx}$), the sector number where it resides currently (S_s), the identity and position of SCA in its sector (IP_s and P_s), and the public key to use in this sector (K_{Ss+}). Hence, the general structure of the NodeRole message sent to a regular node x residing in sector s is:

$$\text{PCA} \xrightarrow{\text{SrcR}} \text{x: [NodeRole, } IP_p\text{, SR, } \{Role_x, Cert_{Nx}, S_s, IP_s, P_s, K_{Ss+}\}\, K_{Nx+}]\, K_{NET-}$$

Referring to Fig. 4, the PCA sends the following NodeRole message to node x, for example:

$$\text{PCA} \xrightarrow{\text{SrcR}} \text{x: [NodeRole, } IP_p, (IP_z, IP_y, IP_x), \{\text{'R'}, Cert_{Nx}, 1, IP_s, P_s, K_{S1+}\}\, K_{Nx+}]\, K_{NET-}$$

where 'R' indicates that x is a regular node and '1' is the current sector of node x. The NodeRole messages are signed using K_{NET-} to assure that the PCA is the message source node. Moreover, private information is encrypted using the public key of node x (K_{Nx+}) to assure that it is the sole node that can decrypt this critical information.

Node x certificate ($Cert_{Nx}$) has the IP address of x (IP_x), the public key of x (K_{Nx+}), a timestamp (t) of certificate creation time, and a time (e) indicating certificate expiration time. These fields are signed together using the K_{NET-}. This certificate is used by node x to authenticate itself to other nodes upon exchanging packets during network maintenance, position acquisition and route initiation stages.

$$Cert_{Nx} = [IP_x, K_{Nx+}, t, e]\, K_{NET-}$$

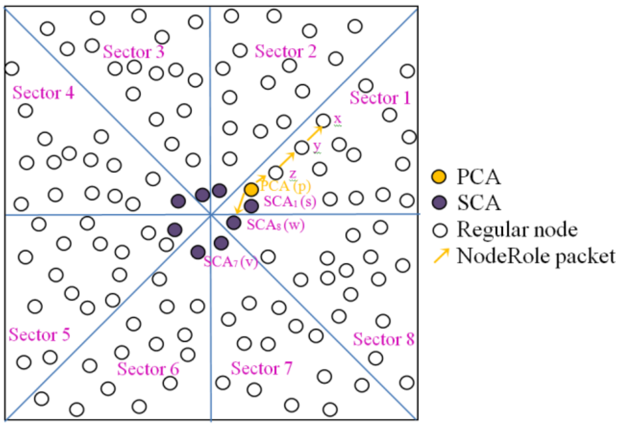


Figure 4. Sending NodeRole packet using source routing

The PCA also unicasts a NodeRole packet for each SCA (w). This NodeRole message holds packet identifier (NodeRole), the source node IP address (IP_P), the source route that the packet should pass through (SR), node w role ($Role_w$), network private key (K_{NET-}), node certificate ($Cert_{Nw}$), the sector number it is responsible for (S_s), sector SCA certificate ($Cert_{Ss}$), private/public key pair to be used within this sector (K_{Ss-}/K_{Ss+}), as well as identity, position and public key of SCA of immediate adjacent sectors (predecessor and successor SCAs in the ring) (Adj_{Ss}). It also contains the Authentication Table (AT) that has a tuple (IP address, public key, certificate time stamp, certificate expire time, and position) of all nodes in this sector. AT is used to update certificates of the nodes. Moreover, it is used when receiving a position request packet; SCA checks if the destination is local or external; to decide whether to send a position reply packet to the source or to send position request packet to successor SCA respectively. Finally, this NodeRole message contains the Adj_{Ns} table that contains the IP address and certificate expiration date for each Node in the predecessor sector of sector s. This table is used during the certificate update process of predecessor sector nodes. The general structure of the NodeRole message sent to node w that will play the role of SCA responsible for sector S_s is:

$$\text{PCA} \xrightarrow{\text{SrcR}} \text{w: [NodeRole, } IP_P\text{, SR, } \{Role_w, K_{NET-}, Cert_{Nw}, S_s, Cert_{Ss}, K_{Ss-}, K_{Ss+}, Adj_{Ss}, AT, Adj_{Ns}\}\, K_{Nw+}]\, K_{NET-}$$

Referring to Fig. 4, node w will receive the following NodeRole message:

$$\text{PCA} \xrightarrow{\text{SrcR}} \text{w: [NodeRole, } IP_P, (IP_w), \{\text{'S'}, K_{NET-}, Cert_{Nw}, 8, Cert_{S8}, K_{S8-}, K_{S8+}, (IP_v, P_v, K_{S7+}; IP_s, P_s, K_{S1+}), AT, Adj_{Ns}\}\, K_{Nw+}]\, K_{NET-}$$

Where ‘S’ indicates that w is a SCA node. Sector s SCA certificate ($Cert_{Ss}$) binds the sector number and its public key and has the sector number, sector public key, time stamp and certificate expire date. These certificates are signed by the sector private key and used by a SCA as an evidence that it is a SCA of the specified sector. These certificates are used between adjacent SCAs and between SCAs and nodes inside their sectors during exchanging network maintenance and position packets. Node w (SCA_8), for example, receives the following sector SCA certificate:

$$Cert_{S8} = [8, K_{S8+}, t, e]\ K_{S8-}$$

B. Network Maintenance

After network setting up stage, nodes may update their certificates, move freely among sectors, and move in and out network borders. Hence, S-Octopus deals with these issues. Each node uses its certificate to apply ARAN authentication steps. Accordingly, any source node signs the packet using its private key and its node certificate is appended to the end of the packet. In case that the source of the packet is an SCA, it also includes its sector SCA certificate to enable the destination to ensure that the SCA has a fresh certificate for a specific sector.

All nodes along the way check their previous node signature (using the public key of the previous node, that is extracted from its certificate), take away the previous node signature and certificate, sign the original contents of the packet, and attach their own certificates.

Another point to be mentioned is that packets sent from the nodes to SCA of their sectors is performed using ResF, as all nodes within that sector know this SCA position. Additionally, communication between nodes (in the same sector or different sectors) is conducted using ResF. ResF is also utilized in communications between adjacent SCAs in neighboring sectors (unless they are reachable within one hop). Though, SrcR is used to send packets from the SCAs to nodes within their sectors; since these SCAs recognize the position of all nodes inside their sectors. Definitely, reply packets are forwarded through RevP of their correlated request packets. Before proceeding further, Table VII summarizes the packets sent during the network maintenance stage.

1) Certifications update

All nodes have to maintain valid certificates with their sector SCA. Nodes periodically send a Certificate Request (CertReq) packet to the SCA of their sector. This CertReq packet is signed with the node private key and is sent by ResF. Fig. 5 illustrates the certificate update packets sent to update node x certificate. Node x sends the following CertReq packet to SCA_4 (node w):

$$x \xrightarrow{ResF} w: [CertReq, IP_w, N_x]\ K_{Nx-}, Cert_{Nx}$$

The CertReq packet contains a packet type identifier (CertReq), SCA IP address (IP_w) and node Nonce (N_x). The packet is signed using the node private key (K_{Nx-}) and the node certificate ($Cert_{Nx}$) is attached to the packet to enable other nodes to verify the signature and ensure the freshness of the certificate.

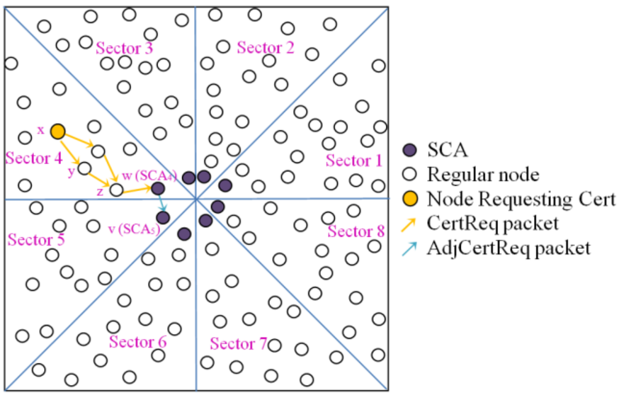
The node nonce is used to uniquely distinguish a packet coming from a specific source. Every time x sends certificate request, it monotonically increases its nonce. So, a particular (IP_x, N_x) pair is used to check whether this CertReq is previously processed. The first node receiving the CertReq packet establishes a reverse path towards the source by storing its IP address. This is due to expecting receiving a certificate reply packet to be sent back to the source node. The neighboring node uses x public key to validate the signature and verify the certificate. The receiving node also ensures that the (IP_x, N_x) tuple of the CertReq has not been already processed. The receiving node adds a new field (Dist) containing the distance between itself and SCA_4, signs the content of the packet, appends its certificate and continues broadcasting the packet. Let y be a neighbor that received the CertReq sent by x. Node y subsequently rebroadcasts the CertReq packet to its one-hop neighbors.

$$y: [[CertReq, IP_w, N_x]\ K_{Nx-}, Dist]\ K_{Ny-}, Cert_{Nx}, Cert_{Ny}$$

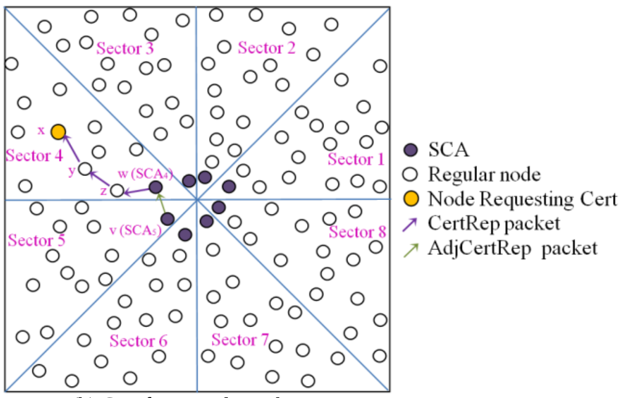
Upon receiving the CertReq packet and using the certificates attached to it, the neighbor z validates the signatures for both x (the CertReq packet initiator) and y (the neighbor it received the CertReq from). Node z now compares the recorded distance (Dist) to the distance between itself and SCA_4 (P_w is not included in the packet since it is known to all nodes inside sector S_4).

If z is closer to SCA_4, it continues sending the packet after modifying the distance in the packet, else it drops the packet. If node z decided to resend the packet, it removes node y certificate and signature, records y as its predecessor, signs the content of the original packet sent by x and appends its own certificate. Then z rebroadcasts the new CertReq packet:

$$z: [[CertReq, IP_w, N_x]\ K_{Nx-}, Dist]\ K_{Nz-}, Cert_{Nx}, Cert_{Nz}$$



(a) Certificate request packets sent



(b) Certificate reply packets sent

Figure 5. Node x certification update.

TABLE VII. PACKETS SENT DURING THE NETWORK MAINTENANCE STAGE OF S-OCTOPUS

	Packet id	Stand for	Description	From	To
Certificate update	CertReq	Certificate Request	• Sent periodically requesting to update the certificate of node x. • Sent using ResF.	Each regular node x	SCA of x
	CertRep	Certificate Reply	• Contains the updated certificate of node x. • Sent through the RevP of the CertReq.	SCA of x	Node x
	AdjCertReq	Adjacent Certificate Request	• Sent to call for the approve to update x certificate. • Sent using one-hop unicast or ResF.	SCA of x	Successor SCA
	AdjCertRep	Adjacent Certificate Reply	• Sent in the case of accepting the certificate update request. • Sent through the RevP of the AdjCertReq.	Successor SCA	SCA of x
Node mobility	DepNode	Departed Node	• Sent when a node x departs to another sector to indicate the trustworthy of x and inform its position. • Sent using one-hop unicast or ResF.	SCA of x	SCA of new sector
	NewSector	New Sector	• Contains the number and public key of the new sector as well as IP address and position of the sector SCA. • Sent using SrcR.	SCA of new sector	Departing node x
	NewNode	New Node	• Contains information about the new node. • Sent using one-hop unicast or ResF.	SCA of new sector	Successor SCA of new sector
	UpdateSPos	Update SCA Position	• Contains the new position of a SCA that has moved d_{mov} from its last known position. • Sent using SecF.	Moving SCA	All nodes in its sector
	UpdateAdjSPos	Update Adjacent SCA Position	• Contains the new position of a SCA that has moved d_{mov} from its last known position. • Sent using one-hop unicast or ResF towards successor and predecessor SCAs.	Moving SCA	Adjacent SCAs
SCA election	SCAElection	SCA Election	• Sent to initiate a new SCA election if a SCA has decided to depart its sector s, or its distance from the network center became higher than a pre-defined distance (d_{cen}). • Sent using SecF.	Departing SCA	All nodes in sector s
	NewSCA	New SCA	• Contains the IP address and position of the new SCA. • Sent using SecF.	Departing (or successor) SCA	All nodes in sector s
	NewAdjSCA	New Adjacent SCA	• Contains the IP address and position of the new SCA. • Sent using one-hop unicast or ResF.	Departing SCA	Adjacent SCAs
Node failure	FailNode	Failed Node	• Contains the IP address and public key of a failed node x to enable it to join the network from any sector. • Sent using one-hop unicast or ResF between successor SCAs till reaching all SCAs.	SCA that issued last certificate for x (if x is a regular node) or successor SCA (if x is a SCA)	All SCAs in the network
SCA synchronization	SysClock	System Clock	• Sent periodically and contains a timestamp to help SCAs keep synchronized clocks. To increase the system robustness, SCAs alternate this job. • Sent using one-hop unicast or ResF between successor SCAs till reaching all SCAs.	Any SCA	All SCAs in the network

All intermediate nodes repeat the same steps as z until the packet reaches SCA_4, which replies to the first CertReq that it receives for a source and a specific nonce. The intended SCA, when receiving the CertReq, communicates its successor SCA to take the permission to update the certificate. This is achieved by sending Adjacent Certificate Request (AdjCertReq) packet to successor SCA. AdjCertReq packet is sent using one-hop unicast if successor SCA is immediate neighbor of sending SCA, otherwise ResF is used. For example, SCA_4 (node w) sends the following AdjCertReq packet to SCA_5 (node v):

$$SCA_4 \xrightarrow{SrcR} SCA_5\text{: [AdjCertReq, } (IP_v), IP_x, N_x, Cert_{S4}] \; K_{Nw-}, Cert_{Nw}$$

AdjCertReq includes a packet type identifier (AdjCertReq), source routing (IP_v) towards SCA_5, sector certificate that SCA_4 has ($Cert_{S4}$) in addition to the IP address (IP_x) and nonce (N_x) of the node requesting the certificate. As with other packets, the sending SCA signs the AdjCertReq packet using its private key (K_{Nw-}) and appends its node certificate ($Cert_{Nw}$). $Cert_{S4}$ is used by the destination SCA (SCA_5) to ensure that SCA_4 has a fresh sector certificate. Fig. 5 (a) illustrates the required certificate request packets (CertReq and AdjCertReq) to update node x certificate.

If the node requesting the certificate is a well-behaving node, successor SCA replies to the AdjCertReq packet by sending an Adjacent Certificate Reply (AdjCertRep). AdjCertRep packets are sent via the reverse paths of their corresponding AdjCertReq packets. For instance, SCA_5 (node v) will send the following AdjCertRep packet:

$$LCA_5 \xrightarrow{RevP} LCA_4\text{: [AdjCertRep, } IP_w, IP_w, IP_x, N_x, Cert_{S5}] \; K_{Nv-}, Cert_{Nv}$$

This AdjCertRep packet includes a packet type identifier (AdjCertRep), IP address of the AdjCertReq packet initiator (IP_w), the IP address of the intermediate node from which node v has received the AdjCertReq packet (IP_w), the sector certificate that LCA_5 possesses

($Cert_{S5}$) in addition to the IP address (IP_x) and nonce (N_x) of the node requesting the certificate. The sending SCA signs the AdjCertRep packet using its private key (K_{Nv-}) and appends its node certificate ($Cert_{Nv}$). SCA_4 is permitted to issue a certificate to the requesting node only upon receiving AdjCertRep packet from its successor SCA (signed by its private key). This technique helps in enhancing protocol robustness and security by avoiding a single point of attack and failure; i.e., an opponent will not be able to issue certificates to untrusted nodes unless it compromised two adjacent SCAs.

In the case of receiving permission from successor SCA, SCA_4 will be able to generate a new certificate for x. Then, SCA_4 unicasts a Certificate Reply (CertRep) packet along the reverse way to the certificate requestor. Let z be the first node that received the CertRep sent by SCA_4:

$$SCA_4 \xrightarrow{RevP} z\text{: [CertRep, } IP_x, IP_z, N_x, Cert_{Nx}, Cert_{S4}] K_{Nw-}, Cert_{Nw}$$

This CertRep packet has a packet type identifier (CertRep), IP address of the original CertReq packet initiator (IP_x), IP address of the intermediate node from which node w received the CertReq packet (IP_z), the sector certificate that SCA_4 possesses ($Cert_{S4}$), the nonce sent by x (N_x) and finally the certificate issued for node x ($Cert_{Nx}$).

Node x uses $Cert_{S4}$ to assure that the SCA issuing the certificate is actually the current SCA for its sector. The sending SCA signs the CertRep packet by its private key (K_{Nw-}) and appends its node certificate ($Cert_{Nw}$).

Nodes receiving the CertRep packet send it back to the neighbor from which they receive the corresponding CertReq. Every node along the reverse path towards the source signs the CertRep and appends its certificate. Fig. 5 (b) demonstrates the forwarded certificate reply packets (AdjCertRep and CertRep).

The SCAs also should keep active node and sector SCA certificates. Thus, periodically each SCA unicasts AdjCertReq to its successor SCA. When receiving the AdjCertRep from the successor SCA, the requesting SCA obtains both node and sector SCA certificates.

2) Nodes mobility

Upon moving a pre-defined distance (d_{mov}) from its last position, a regular node includes its current position in the CertReq packet that it sends to its SCA. This helps the SCAs to stay updated with the positions of the nodes within their sectors and assists them to discover nodes departure to other sectors.

When a node departs to another sector, the SCA of the departed sector removes the node information from its table and sends a Departed Node (DepNode) packet to the successor SCA. DepNode packet informs SCAs that this node is trusted and includes the position of the node. Each SCA in turn, upon receiving this packet forwards it to successor SCA until reaching the intended sector SCA. DepNode packet is sent using one-hop unicast if successor SCA is immediate neighbor of sending SCA, otherwise ResF is used. Fig. 6 shows the conducted communication when x leaves sector S_4 to sector S_8 (moves from position P_x to P'_x). It is clear that SCA_5 (node z) is one-hop from SCA_4 (node y), hence the following one-hop unicast packet is sent:

$$SCA_4 \xrightarrow{SrcR} SCA_5\text{: [DepNode, } (IP_z), 4, 8, IP_x, P'_x, Cert_{Nx}, Cert_{S4}] K_{Ny-}, Cert_{Ny}$$

As a following step, the new sector SCA sends a New Sector (NewSector) packet to the arriving node; including the new sector number and its public key, along with IP address and position of SCA of this sector. This packet is sent using SrcR since SCA is aware of the positions of all nodes in its sector. The new SCA also sends New Node (NewNode) packet to successor SCA informing it about the new node. This packet has the new node IP address and certificate expiration date. This packet is sent using one-hop unicast or ResF. In our example, SCA_8 (node m) sends the following unicast NewSector packet to node x:

$$SCA_8 \xrightarrow{SrcR} x\text{: [NewSector, } (IP_x), 8, K_{S8+}, IP_m, P_m, Cert_{S8}] K_{Nm-}, Cert_{Nm}$$

Also, the following NewNode packet is sent to SCA_1 (node s) for example:

$$SCA_8 \xrightarrow{SrcR} SCA_1\text{: [NewNode, } (IP_S), IP_x, e, Cert_{S8}] K_{Nm-}, Cert_{Nm}$$

If the moving node is a SCA, it sends its position to the nodes within its sector upon moving the pre-defined distance (d_{mov}) from its last identified position. It also sends its new position to SCA of adjacent (successor and predecessor) sectors.

Suppose that SCA_6 (node w) has moved d_{mov} distance from its last known position and that SCA_7 (node V) is within the transmission range of SCA_6. SCA_6 sends the following Update SCA Position (UpdateSPos) and Update Adjacent SCA Position (UpdateAdjSPos) packets to nodes in its sector and its successor SCA (SCA_7) respectively:

$$SCA_6 \xrightarrow{SecF} ALL_6\text{: [UpdateSPos, 6, } P_W, Cert_{S6}] K_{Nw-}, Cert_{Nw}$$

$$SCA_6 \xrightarrow{SrcR} SCA_7\text{: [UpdateAdjSPos, } (IP_V), 6, P_W, Cert_{S6}] K_{Nw-}, Cert_{Nw}$$

where '6' means that SCA of sector 6 is the SCA that has moved to the specified position P_W. UpdateSPos is sent using Sector Flooding (SecF). Hence, any node outside the intended sector discards this packet upon receiving it. Regarding UpdateAdjSPos packet, it is sent using one-hop unicast if successor SCA is immediate neighbor of sending SCA, otherwise ResF is used.

An SCA may decide to leave its sector, or its distance from the network center may exceed a pre-chosen distance (d_{cen}). In both cases, a new SCA election is compulsory. Upon deciding to depart the sector, the SCA sends a SCA Election (SCAElection) packet to nodes within the original sector using SecF. Every node inside the sector calculates its probability and sends this probability to the departing SCA through RevP. The departing SCA choses the highest probability node to play the role of the sector SCA.

Accordingly, the departing SCA sends a New SCA (NewSCA) packet to all nodes within that sector so that they know the identity and position of the selected SCA. This information is also directed to the successor and predecessor SCAs in the adjacent sectors via a New Adjacent SCA (NewAdjSCA) packet. At this point, the departing SCA transfers the needed information to the new SCA (like that included in the NewRole packet sent from PCA to SCA nodes while setting up the network).

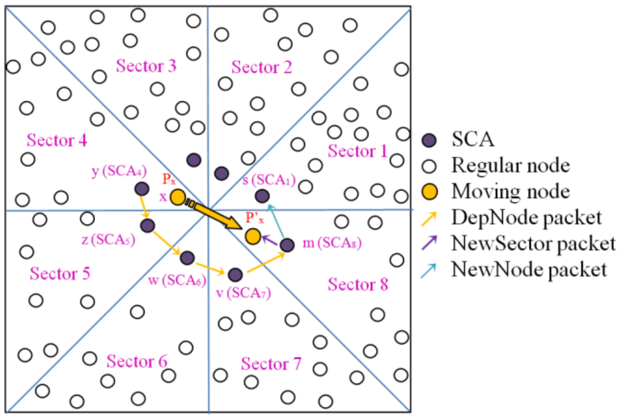


Figure 6. Movement of node x from sector 4 to sector 8.

3) Nodes failure

To mitigate the effect of nodes failure, a SCA backup strategy should be implemented in the system. To maintain an acceptable level of security, each SCA should periodically send a copy of half the information it has to its successor SCA and the other half to its predecessor. The sudden failure of an SCA may be noticed from the periodic SCAs sector certificate update. Thus, if the SCA in a particular sector did not receive the AdjCertReq packet from a specific SCA in a pre-determined time, it will realize that this SCA has a problem. So, successor and predecessor SCAs take the responsibility of choosing a new SCA and sending NewSCA packet.

Afterward, if the failed SCA is repaired, it rejoins the network as a regular node. To make it possible for this node to come back to the network from any sector, its IP address and public key are sent to all sectors SCAs. Hence each SCA sends a Failed Node (FailNode) packet to its successor SCA using one-hop unicast or ResF.

Periodic node certificate update may help in discovering regular nodes failure. If a SCA AT has an expired node certificate, and it has not received a CertReq packet within a specific period, it discovers that this node has a trouble. Accordingly, the SCA that has issued the last certificate for this node will send a FailNode packet.

4) SCAs synchronization

Collaborated SCAs should maintain synchronized clocks to guarantee S-Octopus correctness, i.e.; to avoid issuing certificates with two different time stamps to two nodes in different sectors at the same moment. Hence, there is no need to have the clocks adjusted to a reference clock, instead, nodes refer to their own clocks, while keeping information about the difference between the system clock and their local clocks so that at any time their local time is easily transformed to the system time. At the beginning, PCA includes system time inside the NewRole packet forwarded to SCAs during setting up the network. Thus, all SCAs will be aware of the difference between the system clock and their own clocks. As well, the system time is included within the information sent to a newly elected SCA.

Furthermore, any clock is subject to clock drift; as oscillators frequency may vary unpredictably owing to a variety of physical effects [4]. Thus, periodically one of the SCAs sends a System Clock (SysClock) packet to other SCAs including a time stamp to mitigate the SCAs clocks drift effect. To increase the robustness and distribute load, the SCAs take turn to send this packet considering the anti-clockwise SCAs ring arrangement. SCA includes its sector SCA certificate inside the packet, signs the packet contents, and appends its certificate. These packets are forwarded to all SCAs, using one-hop unicast or ResF between successor SCAs. Supposing that it is SCA_6 (node w) turn to send the SysClock packet. It sends the following packet using source routing to SCA_7 (node v):

$$SCA_6 \xrightarrow{SrcR} SCA_7\text{: [SysClock, }(IP_v)\text{, t, }Cert_{S6}]\ K_{Nw\text{-}},\ Cert_{Nw}$$

Regular nodes take benefit of the timestamp inside their certificates to be aware of the system time and check other nodes certificates validity.

C. Location Service

The source should obtain the position of the destination before starting route discovery process. Source node s sends a Position Request (PosReq) packet to its sector SCA using ResF to enquiry the SCA about the destination d position. Thus, source s in Fig. 7 sends the following PosReq packet to SCA_4 (node w):

$$s \xrightarrow{ResF} SCA_4\text{: [PosReq, }IP_w,\ N_s,\ IP_d]\ K_{Ns\text{-}},\ Cert_{Ns}$$

When receiving the first PosReq, the SCA checks if the destination is in its sector. If the destination is in the same sector, it is found in the SCA AT. Hence the SCA unicasts a Position Reply (PosRep) packet towards the source. This PosRep includes the position of the destination and passes along the RevP to the source. Let the first node receiving PosRep sent by SCA_4 be node v:

$$SCA_4 \xrightarrow{RevP} s\text{: [PosRep, }IP_S,\ IP_v,\ N_S,\ P_d,\ Cert_{S4}]\ K_{Nw\text{-}},\ Cert_{Nw}$$

In case that the destination is in another sector, the destination will not be found in the AT of the SCA. Hence, the SCA sends Adjacent Position Request (AdjPosReq) to its successor SCA in the adjacent sector. Forwarding the AdjPosReq packet to SCAs in the ring is repeated till one of the SCAs (SCA_6 in Fig. 7) locates the destination in its AT. This SCA unicasts an Adjacent Position Reply (AdjPosRep) back along the RevP to source sector SCA. Now source sector SCA unicasts a PosRep back along the RevP towards the source node. All position discovery packets are authenticated by each hop.

Table VIII summarizes the used packets during the location service stage.

TABLE VIII. PACKETS SENT DURING THE LOCATION SERVICE STAGE OF S-OCTOPUS

Packet id	Stand for	Description	From	To
PosReq	Position Request	• Initiated to ask for the position of destination d. • Sent using ResF.	Source node s	SCA of source sector
PosRep	Position Reply	• Contains position of d. • Sent along the RevP of the PosReq.	SCA of source sector	Source node s
AdjPosReq	Adjacent Position Request	• Initiated to ask for the position of destination d. • Sent using one-hop unicast or ResF between successor SCAs.	Source sector SCA	Destination sector SCA
AdjPosRep	Adjacent Position Reply	• Contains position of d. • Sent along the Rev path of the AdjPosReq.	Destination sector SCA	Source sector SCA

D. Route Discovery, Setup and Maintenance

After obtaining the position of the destination, the source starts route instantiation via sending a Route Request (RouteReq) packet using ResF towards the destination node. Thus, source s in Fig. 8 sends the following RouteReq packet to destination node d:

$$s \xrightarrow{\text{ResF}} d\text{: [RouteReq, } IP_d, N_s, P_d] \, K_{Ns-}, Cert_{Ns}$$

Upon receiving the first RouteReq packet, the destination unicasts a Route Reply (RouteRep) packet to the source. The entire route discovery procedure is authenticated hop-by-hop. Assume that the first node receiving the RouteRep sent by d is c:

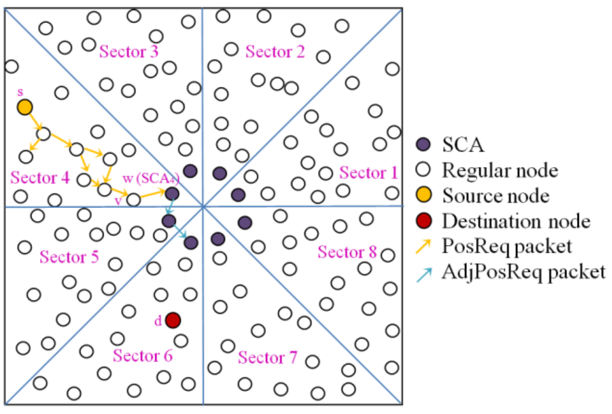
$$d \xrightarrow{\text{RevP}} s\text{: [RouteRep, } IP_s, IP_c, N_s] \, K_{Nd-}, Cert_{Nd}$$

As other on-demand routing protocols, nodes in S-Octopus keep track of routes whether they are still active or not. If no data has been received on an existing route for a specific time, the route is deactivated. Data received on old routes causes nodes to create an Error (Error) packet. Nodes also generate Error packets to announce broken links within active routes.

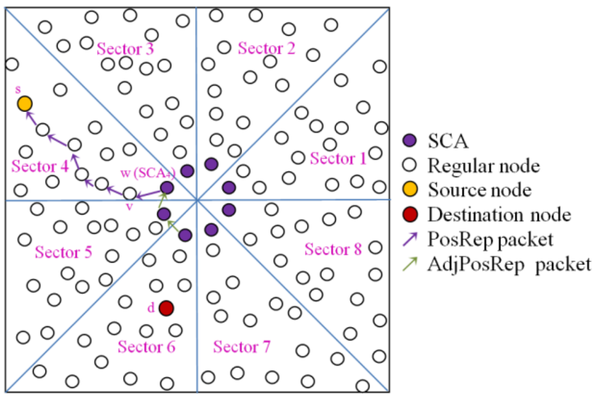
For a route between source s and destination d, a node x, for example, generates the following Error packet:

$$x \xrightarrow{\text{RevP}} s\text{: [Error, } IP_s, IP_d, N_x] \, K_{Nx-}, Cert_{Nx}$$

This packet is signed and forwarded to the source without modification. Nonce (N_x) ensures that the Error packet is fresh. Table IX summarizes the used packets during route instantiation and maintenance stage.

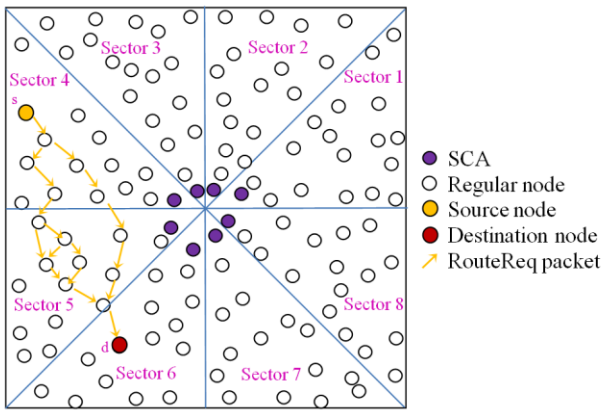


(a) Position request packets sent

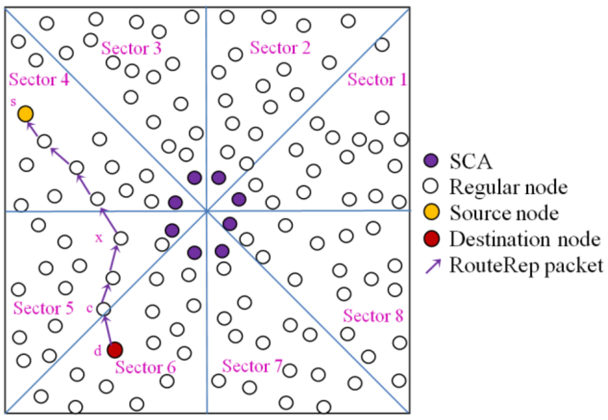


(b) Position reply packets sent

Figure 7. Location service



(a) Route discovery



(b) Route setup

Figure 8. Authenticated route initiation

TABLE IX. PACKETS SENT DURING THE ROUTE INSTANTIATION AND MAINTENANCE STAGE OF S-OCTOPUS

Packet id	Stand for	Description	From	To
RouteReq	Route Request	• Sent to initiate route establishment to destination. • Sent using ResF towards the destination node.	Source	Destination
RouteRep	Route Reply	• Initiated when the destination receives the first RouteReq. • Sent along the RevP of the RouteReq.	Destination	Source
Error	Error	• Generated if data is received on old route or to announce broken links within active routes. • All Error packets must be signed. • Sent to the source without modification.	Node notices the problem	Source

E. Data Transmission

Upon setting up the route, the source initiates transferring data to the destination. As in ARAN and ARANz, only the control packets are signed and verified; once the route reply reaches the source, it is assured that the path is secure. Thus, Data (Data) packets exchanged are no longer processed by S-Octopus. Accordingly, Data packet relaying (RlyD) is used; i.e., nodes simply relay data packets to their successors in the route.

$$s \xrightarrow{RlyD} d: [DATA, IP_s, IP_d, \ldots]$$

F. Misbehavior Detection System

Misbehaving nodes might conduct erratic actions, as using invalid certificates and inappropriately signed packets. S-Octopus responds to all erratic behaviors by dropping the packets showing such behavior. Malicious nodes, however, may cause more severe misbehaving actions and attacks, such as altering some fields in control packets, dropping data packets and fabricating error packets. In these cases, our protocol collaborates with a misbehavior detection system to help in detecting and isolating malicious nodes. Our proposed system is flexible and can be used to protect against several attacks. The main concept is that each node has a Trust Table (TT) to maintain reputation information about neighboring nodes. In the TT, values regarding several events are stored. A node uses this value to evaluate its neighbor as misbehaving (malicious) or well-behaving node. Each node is responsible for gathering information via direct relations and computing its own trust values for its neighbors. Section F.1 discusses the process of collecting data about different trust metrics. After that, dealing with malicious and compromised nodes are explained in Sections F.2 and F.3, respectively.

For comfort of presentation, Table X provides the packets sent during the proposed misbehavior detection system.

TABLE X. PACKETS SENT DURING THE MISBEHAVIOR DETECTION SYSTEM OF S-OCTOPUS

Packet id	Stand for	Description	From	To
MisNode	Misbehaving NODE	• Sent to report misbehaving of other nodes. • Sent using ResF.	Any regular node x	SCA of its sector s
ComNode	Compromised NODE	• Sent after collaboration of the successor SCA of sector s upon receiving a pre-selected number of MisNode messages concerning a specific misbehaving node. • Sent using NetF.	SCA of sector s	All nodes in the network

1) Data collection and trust metrics calculation

One important aspect of trust management systems is collecting data process. Consequently, it is crucial to identify which events provide a valuable feedback to the scheme and assist in making the proper decision. Many trust metrics might be considered to disclose the cooperation willingness of nodes during route establishment and maintenance as well as data forwarding stages. However, as trade-off between security and cost, a subset of these metrics are selected in this work. The behavior aspects that have been chosen for monitoring are:

(1) Control packet modification: nodes gather trust information related to their neighbors during communications about trying to modify some fields in PosReq, PosRep, RouteReq or RouteRep packets.

(2) Data packet dropping: nodes are evaluated concerning their readiness in passing data packets, in a try to mitigate black-hole and grey-hole attacks. Readiness can be checked either based on link layer acknowledgements, or through overhearing [4].

(3) Error packet fabrication: To protect against fabricating Error packets, each node keeps information about the number of Error packets issued by each neighbor.

For the first trust metric, node x calculates trust value regarding node y considering modification attack ($TVMod_{xy}$) using the following equation:

$$TVMod_{xy} = UnModP_{xy} /(UnModP_{xy} + ModP_{xy})$$

Where $UnModP_{xy}$ is the number of unmodified control packets and $ModP_{xy}$ is the number of altered ones received by node x from node y.

For the second trust metric, node x calculates trust value regarding node y considering dropping attack ($TVDrop_{xy}$) using the following equation:

$$TVDrop_{xy} = UnDropP_{xy} /(UnDropP_{xy} + DropP_{xy})$$

Where $UnDropP_{xy}$ stands for the count of delivered data packets and $DropP_{xy}$ is the count of dropped data packets by node y that it already received from node x.

For the last trust metric, node x calculates a trust value regarding neighbor y considering ERR packets fabrication attack ($TVFab_{xy}$) by counting the number of ERR packets issued by y that passes through node x.

2) Malicious nodes

Once $TVMod_{xy}$ or $TVDrop_{xy}$ becomes less than the modification threshold (ThMod) or the dropping threshold (ThDrop) respectively, node x considers node y as a malicious node. Also, if $TVFab_{xy}$ becomes higher than the fabrication threshold (ThFab), node x will be certain that node y is a malicious node. In these cases, node x excludes node y from future communications. Moreover, node x sends a Misbehaving Node (MisNode) packet to report this misbehavior to the SCA of its sector. This packet is sent using ResF. Suppose that the SCA responsible for node x is node s, then node x sends the following MisNode packet to node s:

$$x \xrightarrow{ResF} s\text{: [MisNode, } IP_s, N_x, IP_y] K_{Nx-}, Cert_{Nx}$$

The MisNode packet contains a packet type identifier (MisNode), the SCA IP address (IP_s), the sending node nonce (N_x) and the IP address of the misbehaving node (IP_y). Each time x sends a MisNode packet, its nonce value is monotonically increased. The packet is signed by x private key (K_{Nx-}) and x certificate ($Cert_{Nx}$) is attached to the packet for validation issues.

3) Compromised nodes

If the SCA in a specific sector has received a pre-defined number (Nm) of MisNode messages indicating the misbehavior of a particular node, then SCA broadcasts a Compromised Node (ComNode) packet. Consequently, this node is excluded by other nodes until its certificate expires normally. Supposing that the SCA responsible of the compromised node is node s, then node s broadcasts the following ComNode packet:

$$s \xrightarrow{NetF} ALL\text{: [ComNode, } N_s, [IP_y] K_{NET-}] K_{Ns-}, Cert_{Ns}$$

The ComNode packet is sent using NetF technique; i.e., it reaches each node existing currently in the network. Thus, every node receives this packet continues broadcasting to all its neighbors. The ComNode packet contains a packet type identifier (ComNode), the nonce of the sending node (N_s) and the IP address of the compromised node (IP_y). The packet is signed by s private key (K_{Ns-}) and node certificate ($Cert_{Ns}$) is attached to the packet. To ensure that the node initiated the packet is truly one of the SCAs in the network, IP address of the compromised node is signed by K_{NET-}.

Let us now consider the compromise of SCA nodes. In the case of one compromised SCA or many non-successor compromised SCAs, absolutely the compromised SCA will try not to issue certificates to legal nodes inside its sector. Moreover, its predecessor SCA will not be able to issue certificates to nodes in its sector since it should receive permission from its successor compromised SCA. However, the compromised SCA will still be unable to issue certificates to other untrusted nodes since it should receive permission from its successor SCA.

This state may end by the departure of the compromised SCA to another sector, having the compromised SCA energy run out, receiving the pre-defined number of MisNode packets by the successor SCA indicating the predecessor SCA misbehavior. In these cases, a new SCA election is conducted, by successor SCA to replace the compromised SCA. Having a trusted SCA, it can continue its task as usual. In a worse case, this state might continue till the certificates of all nodes inside the compromised SCA sector and the predecessor sector expire. In this case, these nodes become unable to take part in any activity till the end of the network lifetime.

The worst case happens upon the concurrent compromise of two successor SCAs. At this point, the entire network security is affected and the compromised SCAs are able to cooperate and issue certificates to existing malicious nodes.

IV. Simulated Performance and Security Evaluation

The performance and security of S-Octopus are evaluated and compared with other recent routing protocols. Our protocol is compared with the basic ARAN protocol since our protocol uses the authentication steps proposed by ARAN. In addition, ARANz protocol is considered for comparison issues since it, like our protocol, deals with the network as parts. Hence, a detailed simulated performance and security evaluation of the three routing protocols is provided in this section.

GloMoSim is used to study the performance and security of S-Octopus, ARAN and ARANz protocols. Nodes transmission range of 250m is simulated. Node density of 60nodes/km^2 and random nodes initial positions are simulated. Then, nodes travel based on the random waypoint mobility model, i.e., every node moves to a randomly selected position at a pre-configured speed and then stops for a selected pause time, before selecting another random position and repeating the same steps.

802.11 MAC layer and Constant Bit Rate (CBR) traffic using User Datagram Protocol (UDP) are simulated. Sources and destinations are randomly chosen. In each run, five CBR sessions are conducted. In each session, 1000 data packets of 512 bytes each at the rate of 4 packets per second are generated. Three of the five CBR sessions in each run are chosen to be local and the other two are external; i.e., local communication percentage of 60% has been simulated as the chance for a node to communicate with a nearby node is higher than communicating a distant node.

It has been assumed that the key distribution procedure has been finished. A 16-byte signature and 512-bit key are simulated [12]. For either protocol, a routing packet processing delay of 1ms is configured [5]. Moreover, a processing delay of 2.2 ms is added to account for the cryptographic procedures [12]. In order to minimize collisions, nodes waits a random time between 0 and 10ms preceding the retransmission of a broadcast packet. Each point in the following figures is calculated as the average of five values from five identical configuration simulation runs with different randomly generated values.

The effect of five important *parameters* of MANETs are tested. These parameters are node mobility speed, network size, nodes density, local communication percentage, and malicious node percentage. Six *performance metrics* are evaluated for each parameter; these metrics are:

(1) Packet Delivery Fraction (PDF): portion of data packets created by CBR source nodes and received by their destination nodes.
(2) Average Path Length (APL): average length of the discovered routes by a protocol. It is evaluated by averaging the number of hops taken by different data packets to reach their destinations.
(3) Packet Network Load (PNL): resulted overhead packets from setting up and maintaining network structure together with updating certificates and positions of nodes. In *S*-Octopus and ARANz, it is evaluated as the total packets forwarded during the setup and maintenance stages. Conversely, it is considered in ARAN as the total packets sent to keep active nodes certificates. Each hop transmission is also counted in this metric assessment.
(4) Packet Routing Load (PRL): percentage of routing packets to delivered data packets. Routing packets includes all sent packets in location service, route discovery, route setup and route maintenance steps. The transmission at each hop also is considered in this metric evaluation.
(5) Average Route Latency (ARL): average time needed to discover a route to an intended destination. In ARAN, it is the average time needed by a source to send a route discovery packet and receive the first correlated route reply. In S-Octopus and ARANz, it is the average delay for discovering destination position and initiating a route to it.
(6) Average End-to-End Delay of data packets (AEED): average time spent from sending a data packet by the source till receiving it by the anticipated destination. AEED includes all delays during position inquiry, route construction, intermediate nodes buffering and processing and needed delays at the MAC layer.

Malicious nodes simulate Multi-attack in which they carry out the following *attacks* against data and/or control packets with a specific probability:

(1) Modification attack: Malicious nodes selectively reset the hop count field to 0 in the route discovery and setup packets passing through them. By assigning the hop count field to 0, a malicious node makes other nodes believe that it is just one hop from the source or destination.
(2) Grey hole attack: Misbehaving nodes optionally drop data packets at random times.
(3) Fabrication attack: Malicious nodes conducting this attack periodically fabricate error packets with a specific probability.

For the malicious node percentage scenario, the following *security metrics* have been also studied:

(1) Malicious Route Percentage (MRP): fraction of used routes containing malicious nodes within them. It is evaluated as the number of routes passing through misbehaving nodes divided by the total routes number.
(2) Packet Loss Percentage (PLP): portion of data packets that are dropped by misbehaving nodes.
(3) Fabricated Error Packets (FEP): number of fabricated error messages by malicious nodes.
(4) Compromised Node Percentage (CNP): percentage of nodes that are treated as compromised due to their misbehaving attitude.
(5) Packet Malicious Load (PML): overhead packets for sending misbehaving detection packets including MNODE and CNODE in case of ARANz, and MisNode and ComNode in case of S-Octopus. The transmission at all hops is considered in calculating PML.

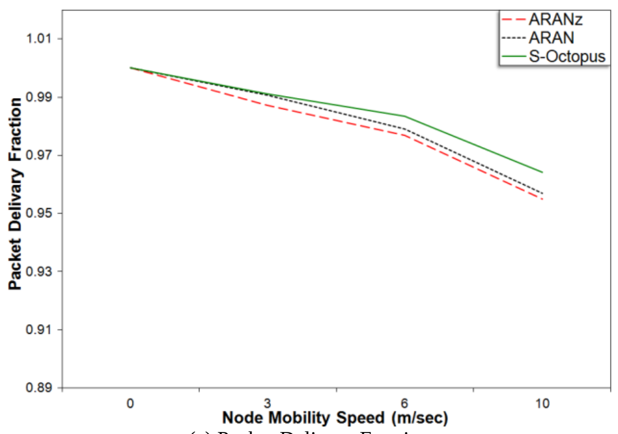
The last two metrics are specified only for S-Octopus and ARANz protocols since ARAN does not have a misbehavior detection system. Some initial experiments have been carried out to choose the best values for modification threshold (ThMod), dropping threshold (ThDrop), fabrication threshold (ThFab) and the MisNode (or MNODE) packets number that should be received by CAs to consider a particular node as compromised (Nm). Different values for Nm are considered ranging from 1 to 3, also ThMod and ThDrop are assigned values ranging from 0.3 to 0.7. Finally, values of ThFab range from 3 to 7. Results of these experiments show that a larger number of malicious nodes are discovered and identified as compromised nodes upon setting Nm, ThMod, ThDrop and ThFab to 1, 0.5, 0.5 and 3, respectively. Accordingly, these are the values that are assigned for these parameters upon simulating different scenarios.

A. Node Mobility Speed Effect

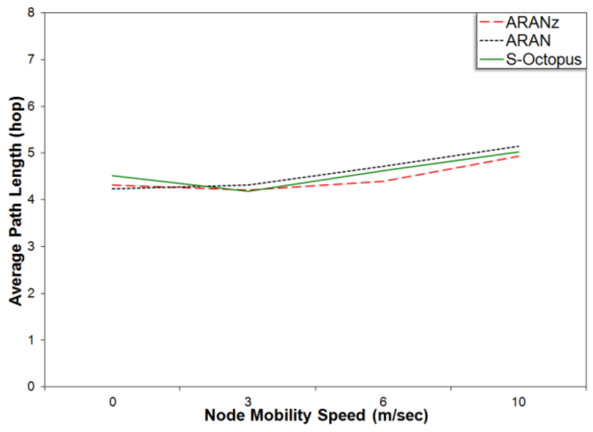
To investigate the node mobility speed effect, a 2km×2km network is simulated. This network has 240 nodes (i.e. 60 nodes/km^2 node density) and it is divided into 9 equal-sized square-shape zones in case of simulating ARANz and into 8 sectors in S-Octopus. Simulations are run with 0m/s, 3m/s, 6m/s and 10m/s speeds with a pause time of 30s. In each run, five CBR sessions are simulated; two of them are external and three are local.

Fig. 9 (a) shows that PDF of ARANz and ARAN is similar to that of S-Octopus in low node mobility, but it is somewhat less upon increasing the mobility. This difference in PDF is related to longer time spent by nodes to process packets sent among LCAs in the case of using ARANz protocol and these sent using broadcast in the case of implementing ARAN protocol. Longer time results in higher probability of losing the link connection due to nodes movement, which in turn results in some packets dropping. In case of using S-Octopus, reduced SCAs communication overhead results in decreasing waiting time required to process data and control packets by each node. Which in turn make the decrease in the PDF slower. It is also obvious from Fig. 9 (a) that PDF for the three protocols decreases a little upon increasing the mobility of nodes. Higher node mobility means higher probability of losing the link connection and dropping some data packets. Moreover, obtained PDF is above 95% in all scenarios using either protocol. This is an evidence that the three studied protocols are effective in discovering routes even upon simulating fast movement of nodes.

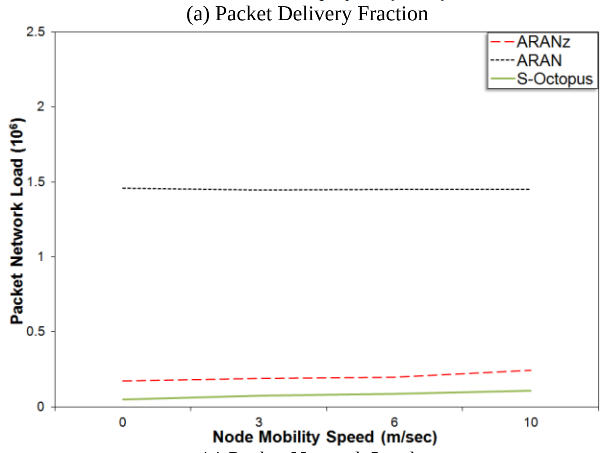
As shown in Fig. 9 (b), the differences in APL of the three protocols are insignificant. APL increases slightly with increasing the node movement. Nodes mobility could result in separating the source and destination from each other and so using longer routes. Fig. 9 (c) shows that PNL of ARANz and S-Octopus are significantly lower than that of ARAN. Nodes in ARAN are unaware of the position of the CA, hence certificate update request packets sent from normal nodes to CA are broadcast to the whole network. In ARANz, however, position and certificate update packets and network structure maintenance packets are sent mainly using ResF or SrcR. S-Octopus has the minimum PNL. This is since all packets required for network structure maintenance as well as certificate and position update are sent between a limited number of SCAs and each SCA sends them only to its successor SCA using one-hop communication in most cases. In ARAN, certificate update request packets are broadcast to the whole network regardless of mobility speed. As such, PNL for ARAN is almost not affected by mobility speed. PNL for ARANz and S-Octopus increases to some extent with increasing the node mobility. Higher node mobility increases the sent packets to update nodes positions and to elect new CAs.



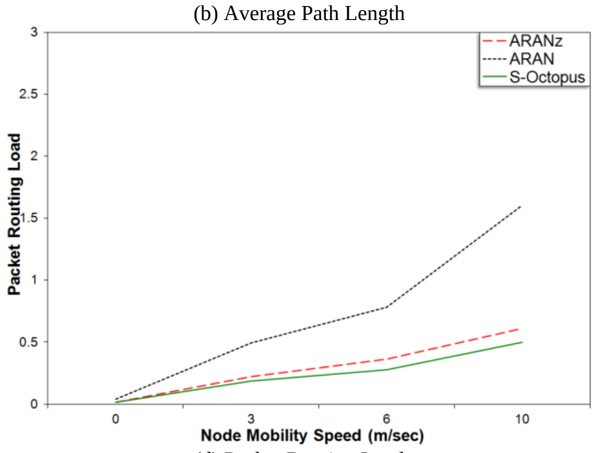
(a) Packet Delivery Fraction



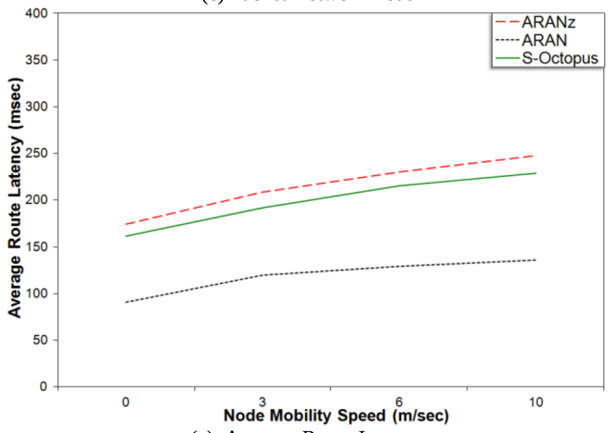
(b) Average Path Length



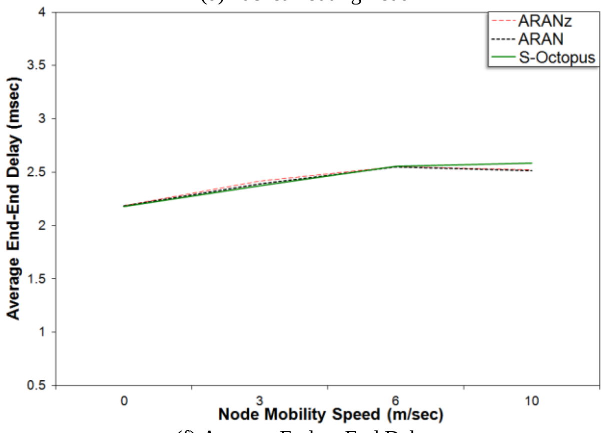
(c) Packet Network Load



(d) Packet Routing Load



(e) Average Route Latency



(f) Average End-to-End Delay

Figure 9. Node mobility speed effect

Fig. 9 (d) shows that ARANz and S-Octopus have lower PRL. ARANz and S-Octopus do not broadcast the RDP packet to the whole area, instead it is sent using ResF towards the destination; hence, the overall PRL is reduced.

Even PDP packets are sent mostly using ResF. However, S-Octopus has lower PRL due to reduced PDP packets especially when the source and destination are in adjacent sectors. Moreover, it is obvious that PRL for the three protocols slightly increases upon increasing mobility speed. Higher mobility increases the chance to lose links and reinitiate RDP packets which increases the value of PRL for the three protocols.

ARANz and S-Octopus have higher latency since they take into consideration the required time to enquiry about the destination position. Fig. 9 (e) shows that S-Octopus has lower ARL compared to ARANz protocol due to reduced time needed for SCAs communications to obtain destination position. Fig. 9 (e) also shows that ARL, of the three protocols, increases as the mobility speed increases; due to having nodes far away from each other which results in longer paths and higher ARL.

If we exclude the needed time to obtain destination position, our results show that ARAN has higher route acquisition latency since ARAN uses broadcast to send the RDP packets; i.e., processing RDP packets of other route discovery processes is delayed until processing this RDP, which increases acquisition latencies of other routes. ARANz and S-Octopus, however, reduces number of processed packets by each node due to using ResF.

The AEED is almost identical in all three protocols (Fig. 9 (f)). Data packets processing at each hop is identical in either protocol since none of them encrypts the data. So, the three protocols have nearly the same average latency for data packets. Moreover, although ARAN and ARANz has higher ARL, the number of position enquiries and route discoveries is negligible compared to the number of delivered data packets, as Fig. 9 (f) shows. Hence, the effect of ARL on AEED of data packets is not noteworthy. Moreover, the results assure that the AEED is roughly not affected by increasing mobility speed.

S-Octopus minimum PNL and PRL, slower decrease in PDF and lower ARL compared to ARANz answer our first research question; i.e., dealing with the network as sectors increases S-Octopus performance compared to the other two protocols.

B. Network Size Effect

To study the network size effect, three networks of 1km×1km, 2km×2km and 3km×3km network sizes are examined. These networks are divided into 9 equal-sized square-shape zones in case of simulating ARANz and into 8 sectors in S-Octopus. Simulations are run with 60 nodes/km^2. Nodes move at a maximum speed of 5m/s and a pause time of 30s. In each run, three local CBR sessions and two external sessions are conducted.

Referring to Fig. 10 (a) it is obvious that PDF for ARAN and ARANz is a little less than S-Octopus, particularly in large area networks. This difference in PDF is due to longer time spent by nodes to process packets sent among LCAs in case of using ARANz protocol and these sent using broadcast in case of ARAN protocol. This longer time results in high probability of losing connection due to nodes movement and some packets dropping. Also, PDF for the three protocols decreases slightly as the network size increases due to the longer paths that the RDP goes through which increases the link break probability and dropping some data packets. However, PDF obtained using either protocol is above 95% for all simulated network sizes, indicating that the three protocols are effective in discovering and maintaining paths even with fairly large network sizes.

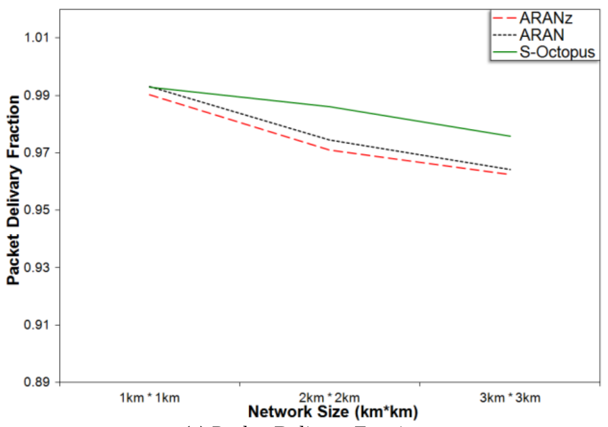
Fig. 10 (b) shows that APL is almost identical, for the three protocols, for a specified network size. This is an indication that the three protocols are efficient in determining the shortest paths. Moreover, APL slightly increases with increasing the network size, due to the longer paths the packet goes through if the source and destination are far away from each other.

As shown in Fig. 10 (c), the PNL for ARAN is significantly larger than that for ARANz. This large gap is due to ARAN certificate update requests broadcast. S-Octopus has the minimum PNL due to reduced SCAs communication overhead. Moreover, PNL for the three protocols increases with increasing the size of the network. Larger networks result in more packets sent for nodes' positions and certificates update due to larger number of nodes existing in the network.

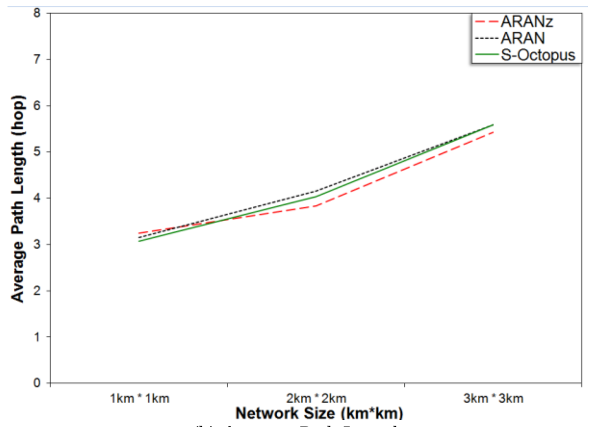
Fig. 10 (d) demonstrates that the PRL increases with increasing the network size due to higher probability of links breakage that needs initiating a new RDP packet. S-Octopus still achieves the minimum PRL due to using ResF in sending RDP packets and reduced PDP packets. On the other hand, ARAN has the highest PRL due to RDP packet broadcast to the entire area.

ARANz and S-Octopus have high latency due to the destination position enquiry time. Fig. 10 (e) shows that S-Octopus has lower ARL compared to ARANz protocol due to reduced time needed for SCAs communications to obtain destination position. ARL increases, for the three protocols, with increasing the network size due to the increased number of nodes (hops) which the control packets pass through. The three protocols produce almost identical AEED values (as shown in Fig. 10 (f)). Although S-Octopus and ARANz have higher ARL, the number of performed route discoveries and position enquiries is small compared to the number of delivered data packets. Hence, the ARL effect on AEED of the data packets is not significant.

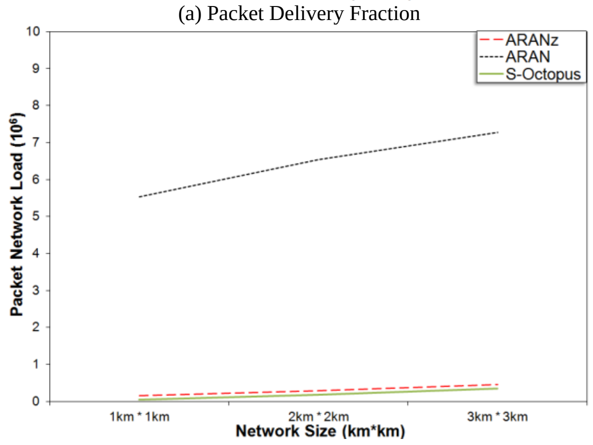
S-Octopus minimum PNL and PRL, maximum PDF and lower ARL compared to ARANz even upon simulating large networks answer our first research question; i.e., S-Octopus is able to achieve better scalability compared to the other two protocols.



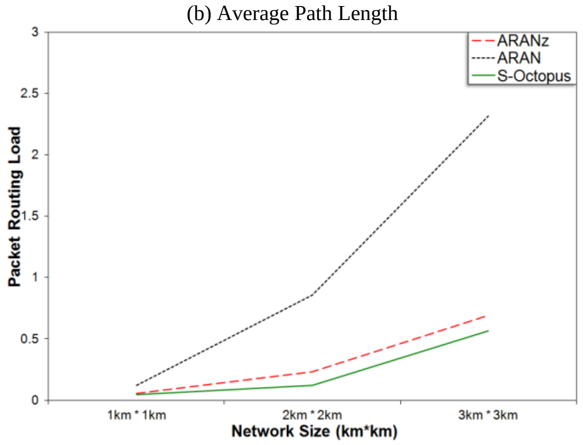
(a) Packet Delivery Fraction



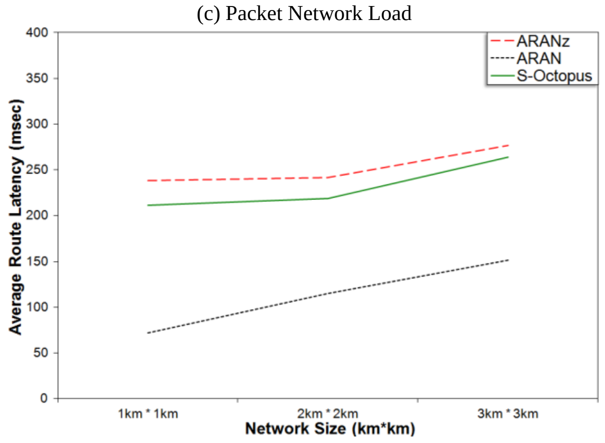
(b) Average Path Length



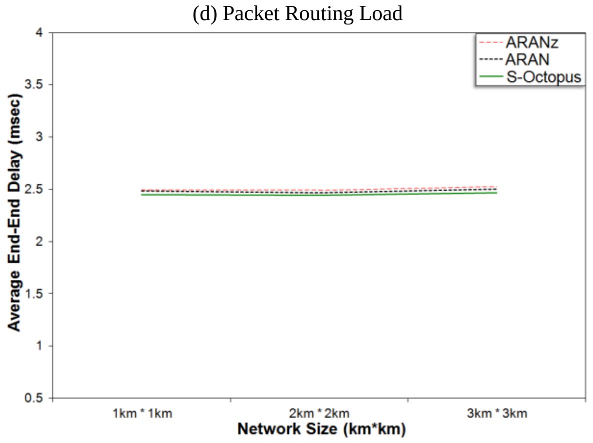
(c) Packet Network Load



(d) Packet Routing Load



(e) Average Route Latency



(f) Average End-to-End Delay

Figure 10. Network size effect.

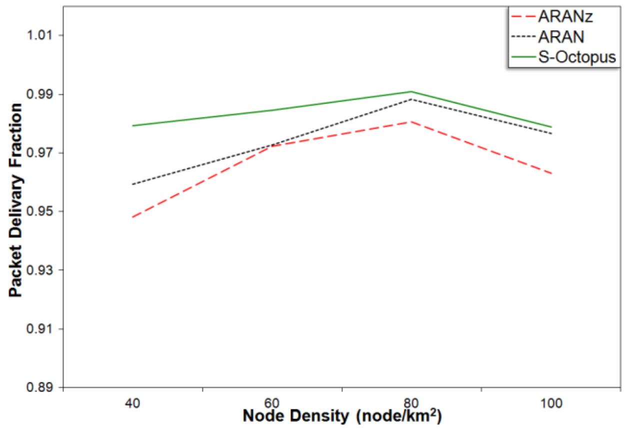
C. Node Density Effect

To test the effect of node density, a 2km×2km network that is divided into 9 equal-sized square-shape zones in case of simulating ARANz and into 8 sectors in S-Octopus is considered. Nodes inside this network move with a 5m/s maximum speed. In each run, five CBR sessions are simulated; three are chosen as local and the other two are external. Experiments are conducted with 40 nodes/km^2, 60 nodes/km^2, 80 nodes/km^2 and 100 nodes/km^2 node densities.

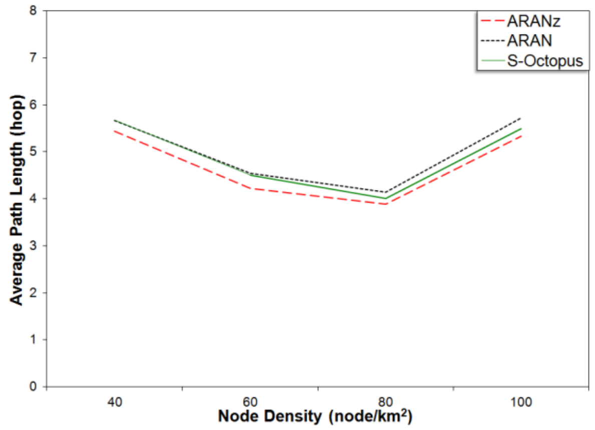
As Fig. 11 (a) reveals, higher PDF for the three protocols is obtained when the simulated node density is between 60 nodes/km^2 and 80 nodes/km^2. On the other hand, upon decreasing density below 60 nodes/km^2, finding a path between the source and destination probability decreases. Furthermore, increasing node density above 80 nodes/km^2 results in increasing the number of nodes participating in rebroadcasting control packets. In other words, an intermediate node receives several copies of the same RDP packet from its neighbors. Processing these control packets may cause delay in processing data packets as well as causing some packet drops. Though, Fig. 11(a) assures that the achieved PDF for the three protocols in all simulated node density values is above 93%. This gives an indication that these protocols discover and maintain routes effectively regardless of network node density.

It is clear from Fig. 11 (b) that the APL decreases with increasing the node density, until reaching its minimum values at node densities ranging from 60 nodes/km^2 to 80 nodes/km^2. This indicates that, as node density increases the chance to find faster path also increases, until reaching 80 nodes/km^2. As density increases above 80 nodes/km^2 APL starts to increase. This indicates that increasing the density more than 80 nodes/km^2 will only make the nodes closer to each other while not serving in finding shorter paths. In fact, increasing the number of control packets received from the neighbors may result in dropping some control packets that may have already passed through the shortest path. However, the differences in APL between the three protocols and for each protocol separately are insignificant. This suggests that the three protocols are efficient in discovering shortest paths irrespective to node density.

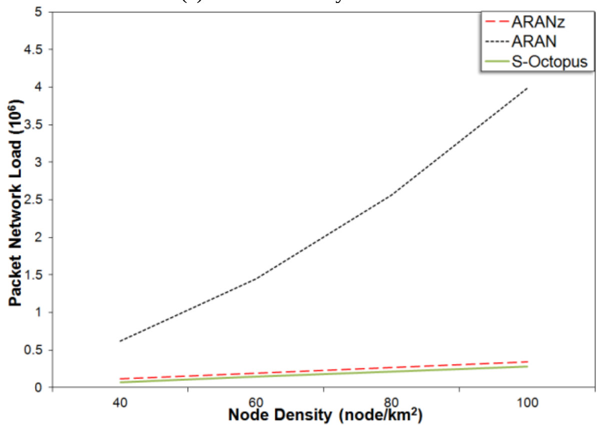
Fig. 11 (c) illustrates that the PNL for ARAN is significantly higher than ARANz since ARAN broadcasts certificate update requests. S-Octopus still has the minimum PNL due to reduced SCAs communication overhead. Moreover, this figure shows that PNL for the three protocols increase with increasing the node density due to increasing the number of nodes updating their certificates and positions. However, the increase in these metrics is more significant upon simulating ARAN protocol. This large gap is a result of ARAN certificate update request packets broadcast to the whole network.



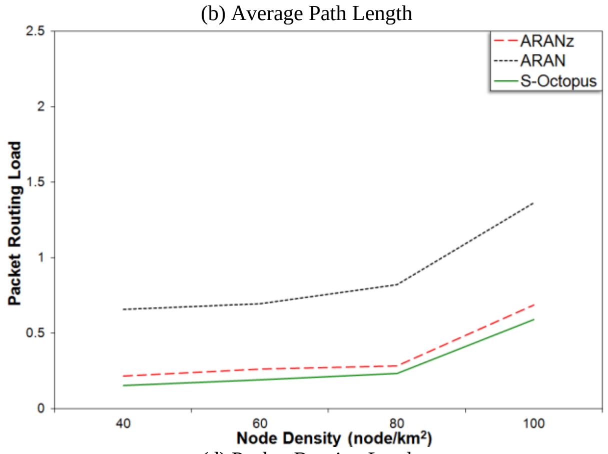
(a) Packet Delivery Fraction



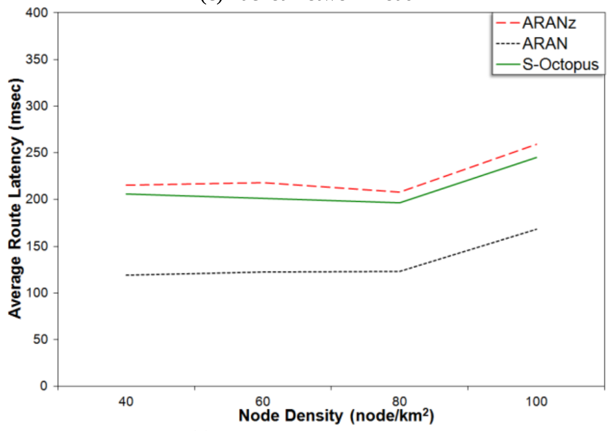
(b) Average Path Length



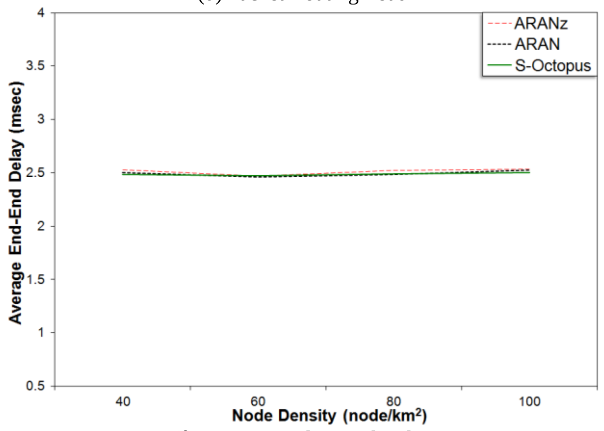
(c) Packet Network Load



(d) Packet Routing Load



(e) Average Route Latency



(f) Average End-to-End Delay

Figure 11. Node density effect.

It is conspicuous from Fig. 11 (d) that the PRL slightly increase with increasing the node density, due to additional nodes receiving and broadcasting RDP packets. S-Octopus still achieves the minimum PRL because of using ResF in sending RDP packets and reduced PDP packets.

Fig. 11 (e) demonstrates that the ARL increase as node density increases. This increase is a result of the increased number of nodes participating in forwarding RDP packets, which causes congestion as well as delay in processing control packets. Fig. 11 (f) assures that AEED is almost identical for the three protocols and is not affected by increasing node density.

S-Octopus minimum PNL and PRL, maximum PDF and lower ARL compared to ARANz confirm our first research hypotheses; i.e., dividing the network into sectors enhances S-Octopus performance compared to the other two protocols.

D. *Local Communication Effect*

To evaluate the effect of local communication percentage, a 2km×2km network which is divided into 9 zones in case of simulating ARANz and into 8 sectors in S-Octopus is considered. A total of 240 nodes are arbitrarily placed in this network. The nodes move at 5m/s speed. Five CBR sessions are performed in each run. Simulations are run with 0%, 40%, 60% and 100% local communication.

As revealed in Fig. 12 (a), PDF obtained using either protocol slightly increases as the percentage of local communication increases and nearly reaches 100% when all communications are local. Larger percentage of local communications means shorter paths, i.e. lower probability of having link breakage and data packet drops. Moreover, it is clear from the figure that the obtained PDF for either protocol is above 96% in all scenarios; indicating that these protocols discover and maintain routes effectively.

Fig. 12 (b) shows that the three protocols are efficient in discovering the shortest paths regardless of the simulated local communication percentage. The same figure indicates that APL decreases for all protocols with increasing local communication because the source and destination are closer to each other.

Fig. 12 (c) shows that the PNL is not affected by local communication percentage because the packets sent for updating nodes certificates and maintaining network structure are sent regardless of the number and type of communication sessions among nodes. Fig. also shows that PNL for ARANz and S-Octopus is still much less than this for ARAN.

Fig. 12 (d) reveals that the PRL curves for the three protocols slightly decrease upon increasing the local communication as a result of the shorter paths. Shorter paths decrease the probability of link break, which in turn, reduces the need for reinitiating a new RDP packet. S-Octopus still has the smallest PRL due to using ResF in sending RDP packets and reduced PDP packets.

As expected, Fig. 12 (e) shows that ARANz has the highest ARL because ARANz needs to carry out a position discovery step and has LCAs communication overhead. However, ARANz ARL improves as more and more packets become internal ones because the nearest LCA, upon receiving a PDP packet, will find the destination in its authentication table, so there is no need to communicate LCAs in other zones.

In fact, all protocols have better ARL as more packets are delivered locally due to shorter paths although ARL curve of ARAN decreases at a slower rate compared to ARANz and S-Octopus. The reason behind this gap is that the RDP packets in ARAN are flooded to the whole area even if the communications are local. This flooding affects ARL for other external communications by increasing the processing delay of other RDP packets.

Fig. 12 (f) shows that AEED curves for the three protocols are almost identical. Although S-Octopus and ARANz have higher ARL, the number of performed route discoveries and position enquiries is small; thus, the ARL effect on AEED is insignificant. Fig. 12 (f) also demonstrates that AEED slightly decreases with increasing local communication due to the shorter paths whether for data or control packets.

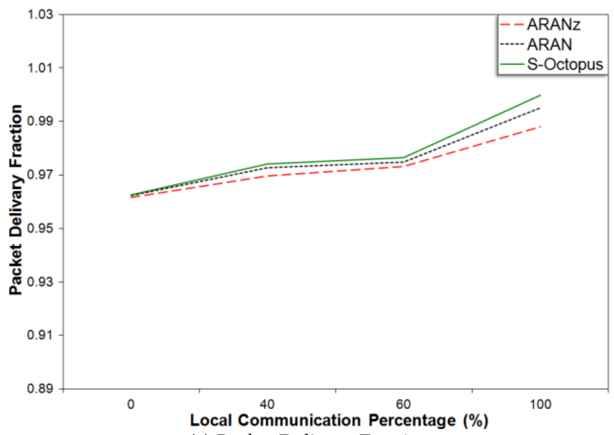
S-Octopus minimum PNL and PRL besides lower ARL compared to ARANz confirm that dealing with the network as sectors help S-Octopus to attain enhanced performance regardless the percentage of local communications.

E. *Malicious Node Percentage Effect Considering Multi-Attack*

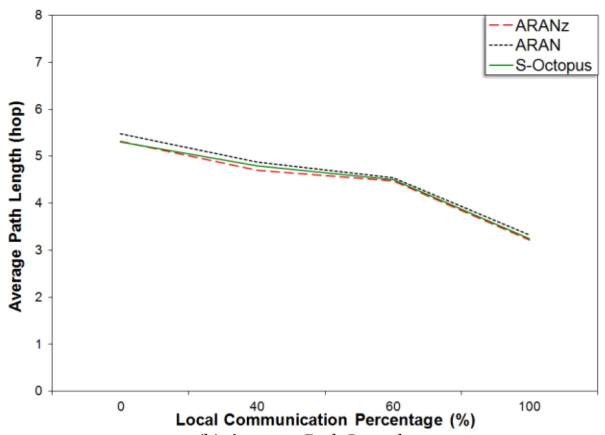
The malicious node behavior effect is tested on a 2km×2km network which is divided into 9 equal-sized zones in case of simulating ARANz and into 8 sectors in S-Octopus. A total of 240 nodes are randomly placed in this network. These nodes move at a maximum speed of 5m/s. Three local sessions and two external sessions are simulated. Simulations are conducted with randomly chosen 0%, 10%, 20% and 40% malicious nodes. Malicious nodes perform multiple attacks with a specific probability. To simulate multi-attack, malicious nodes perform modification, grey hole and fabrication attacks. Malicious nodes performing multi-attack illegally reset the hop count field to 0 in a received route discovery or route reply, if a drawn number is less than 0.5 (modification attack). To perform grey hole attack, malicious nodes drop a received data packet if a drawn number is less than 0.5. Moreover, they conduct fabrication attack by periodically fabricating ERR packet if a chosen number is less than 0.5.

Referring to Fig. 13 (a) it is clear that increasing malicious node percentage results in decreasing PDF for all protocols. This is mainly due to data packets dropped upon performing grey hole attack. The slower decrease in ARANz and S-Octopus PDF is an indication that they are effective in identifying and isolating multi-attack malicious nodes even if the simulated percentage is large.

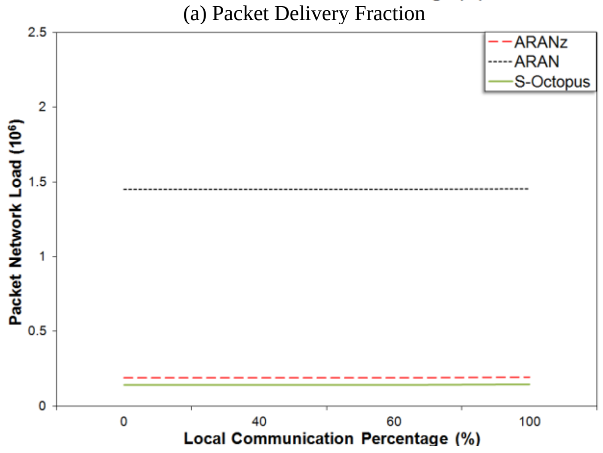
Malicious nodes can exploit non-secure protocols via modification attack, so that non-optimal paths are selected. Fig. 13 (b) shows that the three evaluated secure protocols are not exploitable in this way. The chosen route might go through a malicious node but not compulsory to do so.



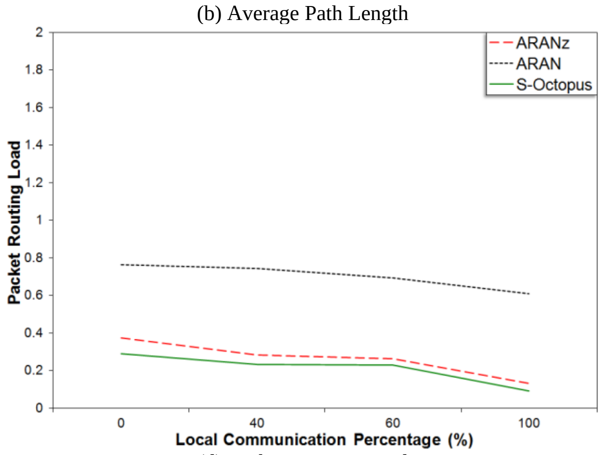
(a) Packet Delivery Fraction



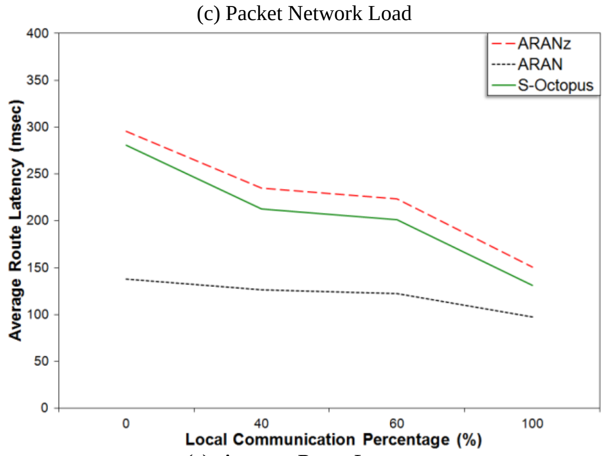
(b) Average Path Length



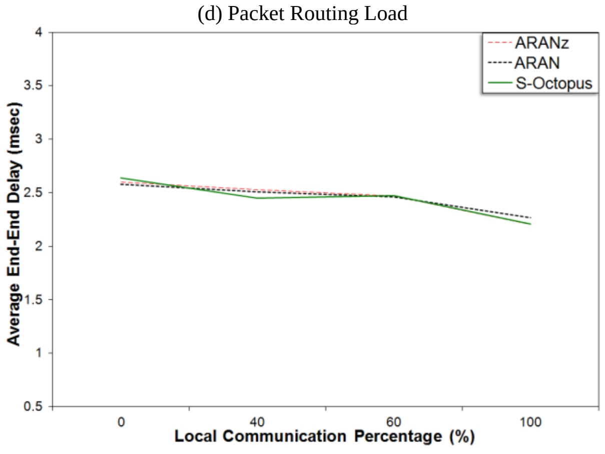
(c) Packet Network Load



(d) Packet Routing Load



(e) Average Route Latency



(f) Average End-to-End Delay

Figure 12. Local communication effect

It is conspicuous from Fig. 13 (c) that malicious node percentage definitely does not affect PNL for the three protocols. The reason behind the stable PNL is that updating nodes' certificates and positions is carried out regardless the number of existing malicious nodes.

Fig. 13 (d) shows that PRL increases with increasing malicious node percentage. This increase in PRL is mainly due to reinitiating RDP packets by the source upon receiving the fabricated ERR packets. Also, it is apparent that ARANz and S-Octopus have the minimum PRL and the slowest increase in PRL, which reflects their effectiveness in detecting and isolating the fabrication attackers.

Fig. 13 (e) shows that ARL for the studied protocols is not affected by increasing malicious node percentage since they are robust against modification attacks.

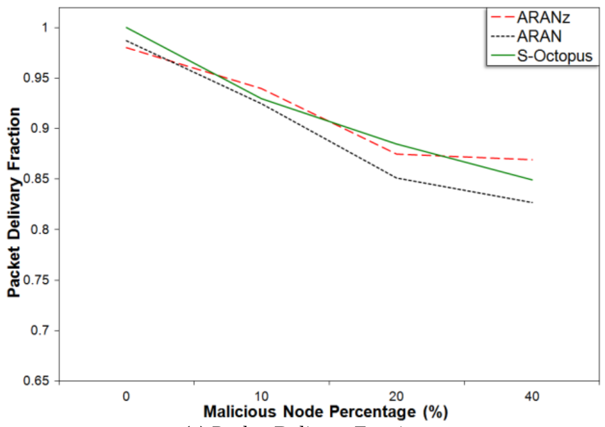
Fig. 13 (f) demonstrates that AEED curves for the three protocols are almost identical. Additionally, increasing malicious node percentage does not affect the AEED of the evaluated protocols.

S-Octopus minimum PNL and PRL in addition to lower ARL approve our second research hypotheses; i.e., using the proposed misbehavior detection system helps S-Octopus to attain enhanced performance even with large percentage of malicious nodes.

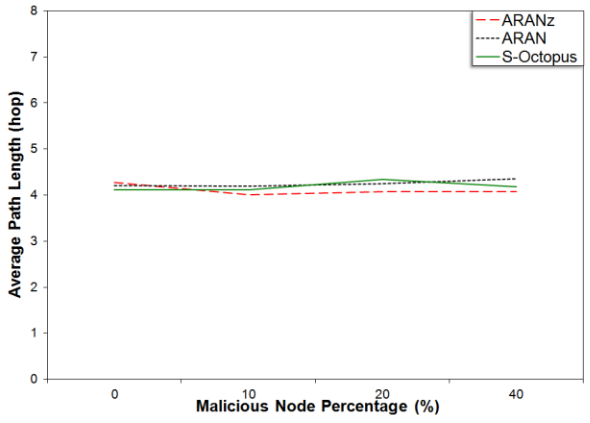
Upon resetting the hop count field to 0, the malicious node forces non-secure protocols to select the route passes through itself if the protocol selects the shortest path. The three simulated protocols, on the other hand, are not exploited in such way. As shown in Fig. 13 (g), the MRP slightly increases for the three protocols as the malicious node percentage increases since the selected routes might go through a misbehaving node. It is clear from Fig. 13 (h) that the PLP of the evaluated protocols increases upon increasing malicious node percentage due to dropping data packets via the grey hole attack. However, upon using ARANz and S-Octopus, the increase in PLP is significantly slower indicating that both of them are efficient in isolating grey hole attackers.

The FEP for the three protocols increases upon increasing the malicious node percentage (as shown in Fig. 13 (i)). Though, the increase in FEP is much slower upon using ARANz and S-Octopus, which illustrates that both of them are effective in extracting nodes performing fabrication attack.

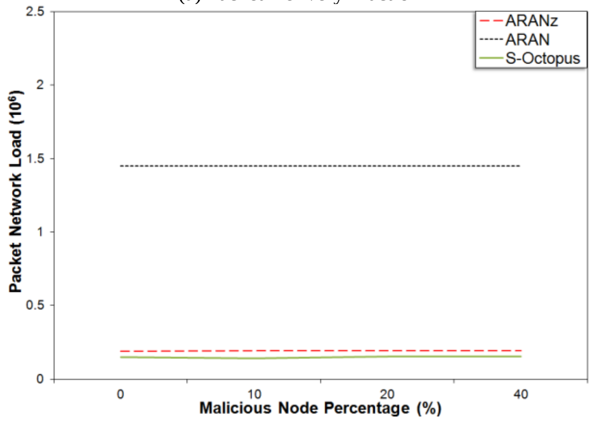
Fig. 13 (j and k) show that as malicious node percentage increases, both ARANz and S-Octopus demonstrate their effectiveness in detecting more and more malicious nodes, i.e. CNP and PML highly increase upon increasing the number of malicious nodes performing multi-attack. This implies that S-Octopus is efficient in discovering malicious nodes and confirms our second research hypotheses; i.e., applying the misbehavior detection system assures S-Octopus security regardless malicious nodes percentage.



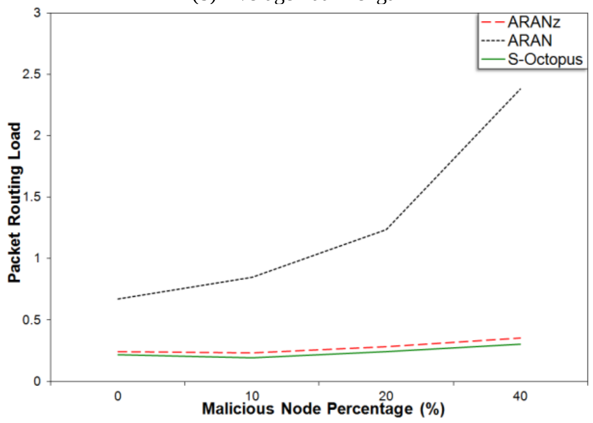
(a) Packet Delivery Fraction



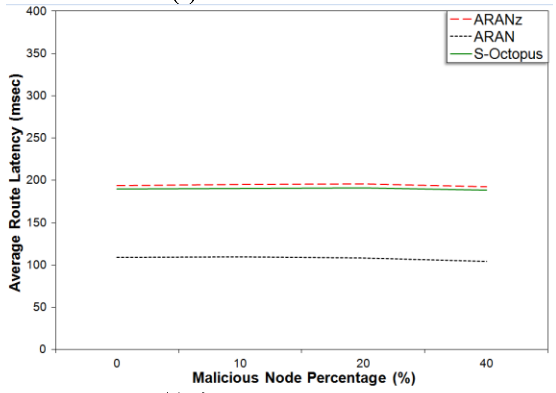
(b) Average Path Length



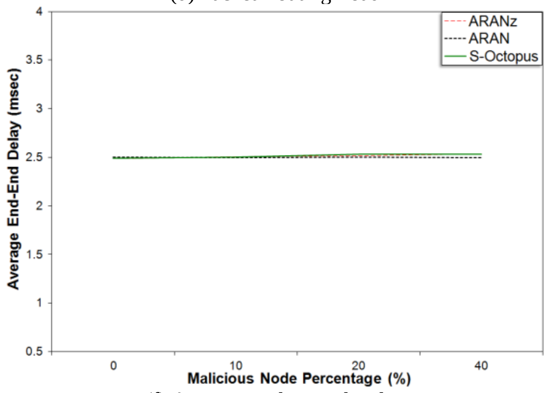
(c) Packet Network Load



(d) Packet Routing Load

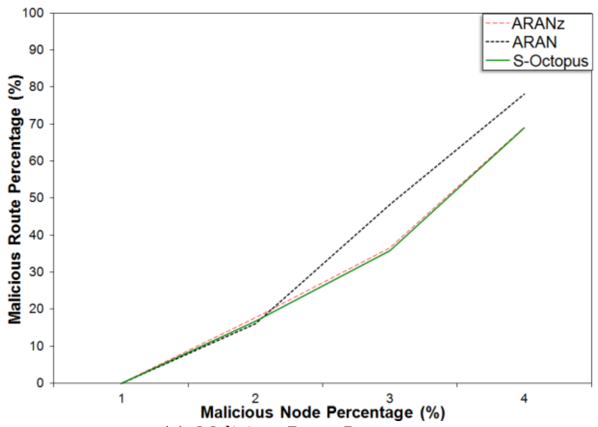


(e) Average Route Latency

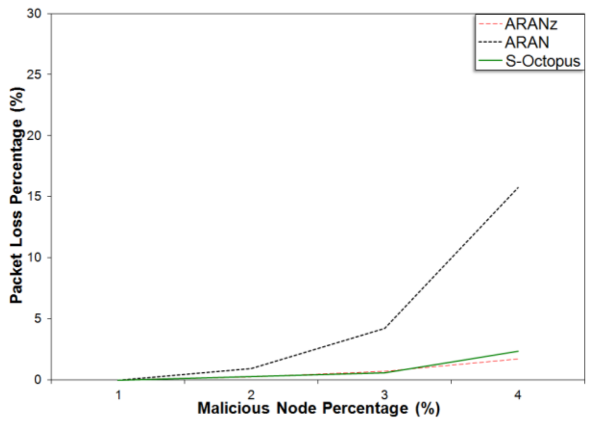


(f) Average End-to-End Delay

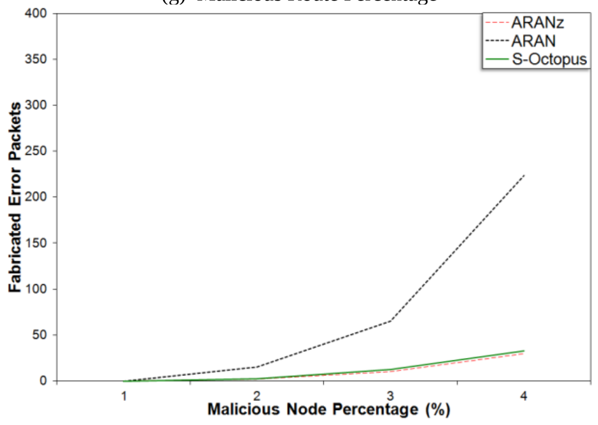
Figure 13. Malicious node percentage effect considering multi-attack.



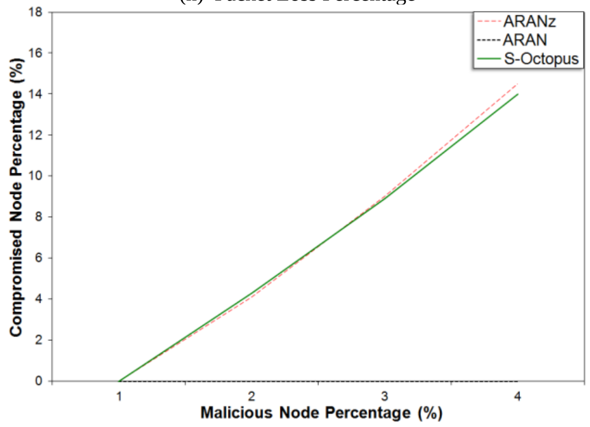
(g) Malicious Route Percentage



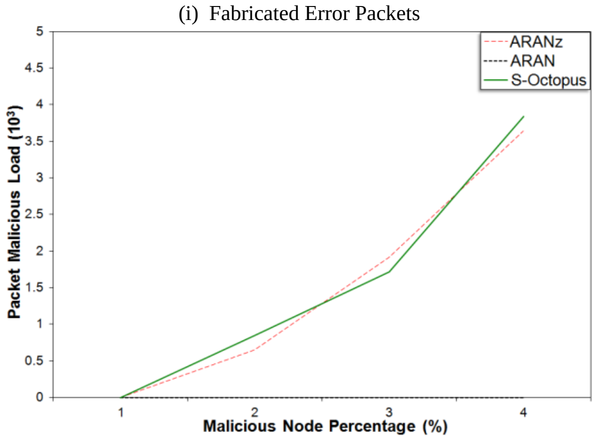
(h) Packet Loss Percentage



(i) Fabricated Error Packets



(j) Compromised Node Percentage



(k) Packet Malicious Load

Figure 13. Malicious node percentage effect considering multi-attack (continued)

V. Results Summary and Discussion

From the obtained simulation results, presented in the previous section, many points are concluded. These points are summarized as follows:

(1) PDF is above 95% in most scenarios, indicating that the three protocols are extremely effective in discovering routes even with fairly large area networks and high node mobility while considering different node densities and different local communication percentages. Upon studying the effect of malicious node percentage, results show that the decrease in PDF is much slower in S-Octopus in most cases. Moreover, S-Octopus is still able to maintain the highest PDF, implying that S-Octopus is efficient in detecting malicious nodes even with relatively large percentage of them.

(2) PNL for S-Octopus and ARANz is significantly smaller than ARAN. A key cause to this gap is that nodes in ARAN are not conscious of the location of the CA server, thus, all sent certificate update request packets towards CA are broadcast to the whole network. In ARANz, however, most packets are sent using restricted directional flooding, source routing, zone flooding or LCA flooding. S-Octopus has the minimum PNL in all scenarios. This is since all packets required for network structure maintenance and certificate and position update are sent between a limited number of SCAs and each SCA sends them

only to its successor SCA using one-hop communication in most cases.

(3) S-Octopus has the minimum PRL in all experiments. In contrast to ARAN, ARANz and S-Octopus do not broadcast the route discovery packets to the whole area, instead, these packets are forwarded using restricted directional flooding to the destination. Even position discovery packets in ARANz are sent via restricted flooding or source routing. Thus, position discovery packets do not affect PRL significantly, especially in case of local communications. Moreover, S-Octopus has lower PRL due to reduced position discovery packets especially if the source and the destination are in adjacent sectors.

(4) S-Octopus highest PDF, lowest PNL and PRL in addition to lower ARL compared to ARANz approve our first research hypotheses; i.e., S-Octopus is able to attain enhanced performance and scalability compared to the other two protocols.

(5) APL is almost identical for the three protocols for the same parameters setting. In other words, even though these protocols do not clearly search for the shortest paths, the first RDP packet usually passes along the shortest path. Thus, it is noticeable that the three protocols are efficient in discovering shortest routes.

(6) ARANz and S-Octopus have high latency because they need to carry out a destination position discovery step. However, S-Octopus has lower ARL compared to ARANz protocol due to reduced time needed for SCAs communications to obtain destination position especially if source and destination nodes are in adjacent sectors.

(7) Differences in AEED between the three protocols are almost negligible since the number of route discoveries and position enquiries performed is limited compared to the amount of delivered data packets. Accordingly, the effect of ARL on AEED of data packets is not noteworthy.

(8) Increasing node mobility and network size and decreasing local communication percentage result in decreasing PDF and increasing PRL and ARL. However, PDF for the three protocols is above 95% in most scenarios assuring that the three protocols are highly effective in discovering routes and delivering data packets.

(9) High PDF and low APL for all protocols are obtained for node density values between 60 nodes/km^2 and 80 nodes/km^2. However, PDF for all protocols is above 94% for all simulated node density values. This suggests that the three protocols are efficient in discovering the shortest paths regardless of node density.

(10) Increasing malicious node percentage results in decreasing PDF and increasing MRP, PLP and FEP for the three protocols. In most cases, however, the decrease or increase in these metrics is slower upon using S-Octopus and ARANz. This suggests that both protocols are efficient in isolating malicious nodes.

(11) Moreover, as malicious node percentage increases, S-Octopus and ARANz effectiveness in distinguishing and isolating malicious nodes is increasingly demonstrated by achieving higher CNP. Both protocols are efficient in identifying and isolating malicious nodes performing modification attack against control packets, grey hole attacks against data packets and ERR packets fabrication attack. Discovering malicious nodes and excluding them from future routes may result in reinitiating route discovery packets and choosing non-optimal paths that do not include malicious nodes within them, hence, causing higher PML and PRL. This implies that S-Octopus is efficient in discovering malicious nodes and confirms our second research hypotheses; i.e., utilizing the proposed misbehavior detection system assures S-Octopus security, while attaining improved performance, regardless the percentage of malicious nodes existing in the network.

To summarize, the conducted experiments prove the efficiency of the simulated protocols in discovering and conserving shortest paths. The results suggest that S-Octopus has realized the scalability issue by preserving the maximum packet delivery fraction and the minimum network and packet routing loads even within large networks and highly moving nodes. S-Octopus scalability is a typical result of applying restricted flooding to send route discovery packets along with the fact that all network structure maintenance, certificate update and position update packets are sent between a limited number of SCAs and each SCA sends them only to its successor SCA using one-hop communication in most cases.

On the other hand, ARANz and S-Octopus have higher route acquisition latency due to time required to obtain destination position. However, this time for S-Octopus is lower compared to ARANz due to reduced time needed for SCAs communications to obtain destination position. Finally, utilizing the misbehavior detection scheme assisted S-Octopus to guarantee high-level of security by recognizing and isolating nodes conducting malicious attacks. Hence, S-Octopus can be a suitable choice for MANETs established among peers at a conference or students on a campus, where it is possible to pre-deploy some keys and certificates.

VI. CONCLUSION

The nature of MANETs makes finding an efficient and secure routing an important issue. AODV is an unsecure protocol resulting in low processing overhead. On the other hand, AODV broadcasts route discovery packets which increases packet overhead. Consequently, AODV is considered as unscalable protocol. ARAN also sends the route discovery packet to all nodes in the network, while utilizing cryptographic certificates to detect bad actions. However, using these certificates results in higher route acquisition latency and higher processing and packet overhead. The centralized trust is considered another issue in ARAN.

ARANz proposes a hierarchal algorithm to improve performance and scalability through dealing with the area as zones. Using numerous LCAs achieved robustness and enhanced security. ARANz also exhibited improved scalability and performance through using restricted directional flooding. However, using several LCAs in ARANz, comes up with the need to synchronize them. Moreover, dividing the network into several square-shaped zones and introducing multiple LCAs on the boundaries of these zones, resulted in high control overhead and delay in accomplishing the operations that require communication among LCAs such as updating nodes certificates and positions, obtaining destination position, LCAs synchronization, and announcing malicious or compromised nodes.

A novel routing protocol, S-Octopus, is suggested in this paper. S-Octopus comes to mitigate ARAN and ARANz problems. S-Octopus avoids the single point of attack and failure problems associated with ARAN by announcing several SCAs. It also solves ARANz problems by dividing the area into sector-shaped regions and selecting several SCAs to be as close as possible to the center of the network. This helps S-Octopus to achieve higher scalability by reducing the resulted SCAs communication overhead, which in turn reduces overall packet overhead and packet processing latency. S-Octopus also aims to achieve improved scalability, performance, and robustness through the restricted directional flooding position-based scheme.

A detailed performance and security evaluation of S-Octopus, ARAN and ARANz protocols have been conducted. The obtained results assure that our research hypotheses have been approved:

(1) S-Octopus accomplished the scalability issue by preserving the maximum packet delivery fraction and the minimum network and packet routing loads even with fairly fast node mobility, large network area, different node densities, and different local communication percentages.
(2) S-Octopus is able to have superior performance and security even with having large percentage of malicious nodes conducting modification, grey hole and fabrication attacks.

ARANz and S-Octopus have higher route acquisition latency due to time required to obtain destination position. However, this time for S-Octopus is lower compared to ARANz due to reduced time needed for SCAs communications to obtain destination position especially if source and destination nodes reside in adjacent sectors.

Consequently, the obtained results approve that S-Octopus together with the proposed misbehavior detection system have achieved performance, scalability and security. Hence, S-Octopus is considered a right choice for managed-open networks connecting students on a campus or employees in a factory; since the pre-deployment of keys and certificates is probable. Furthermore, the proposed misbehavior detection system can be incorporated into other existing non-secure routing protocols.

VII. Future Works

This work is just a starting point for further research. Firstly, more investigation is required to evaluate S-Octopus performance and security. For instance, S-Octopus performance can be investigated under different traffic generation applications, different mobility models, or when nodes are not evenly geographically distributed.

Secondly, S-Octopus may be modified to deal with different number of sectors. Moreover, dynamic and adaptive routing protocols are exciting topics, in which, some routing protocol details might be changed in view of the present state of the network. Regarding the misbehavior scheme, it may be enhanced to identify other types of attacks. More consideration can be given to authentication and key distribution.

Thirdly, one of the significant research limitations facing MANETs is the real environment implementation and testing particularly when the number of nodes is large. So, we aim to implement our protocol via real application.

Finally, this paper concentrated on one of the central MANET concerns; i.e., security issue. However, there are still various open research challenges facing MANETs such as Quality-of-Service and energy-efficiency.

Conflict of Interest

The author declares no conflict of interest.

References

[1] L. Huy, L. Ngoc, and N. Tam, "AOMDV-OAM: A security routing protocol using OAM on mobile ad hoc network," *Journal of Communications*, vol. 16, no. 3, pp. 104–110, 2021.

[2] F. Khan, A. Khan, S. Khan, I. Qasim, and A. Habib, "A secure core-assisted multicast routing protocol in mobile ad-hoc network," *Journal of Internet Technology*, vol. 21, no. 2, pp. 375–383, 2020.

[3] L. Qabajeh, M. Mat, and M. Qabajeh, "A scalable and secure position-based routing protocol for ad-hoc networks," *Malaysian Journal of Computer Science*, vol. 22, no. 2, pp. 99-120, 2009.

[4] L. Qabajeh, M. Mat, and M. Qabajeh, "A more secure and scalable routing protocol for mobile ad hoc networks," *Security and Communication Networks*, vol. 6, no. 3, pp. 286–308, 2013.

[5] C. Perkins and E. Royer, "Ad hoc on-demand distance vector routing," in *Proc. IEEE 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, New Orleans, LA, 1999.

[6] J. Maxa, M. Mahmoud, and N. Larrieu, "Performance evaluation of a new secure routing protocol for UAV ad hoc network," in *Proc. IEEE/AIAA 38th Digital Avionics Systems Conference (DASC)*, pp. 1–10, San Diego, United States, 2019.

[7] A. Prasanth and S. Pavalarajan, "Implementation of efficient intra- and inter-zone routing for extending network consistency in wireless sensor networks," *Journal of Circuits, Systems and Computers*, vol. 29, no. 8, pp. 1–25, 2020.

[8] S. Kalime and K. Sagar, "A review: Secure routing protocols for mobile adhoc networks (MANETs)," *Journal of Critical Reviews*, vol. 7, no. 19, pp. 8385–8393, 2020.

[9] L. Qabajeh and M. Qabajeh, "Detailed security evaluation of ARANz, ARAN and AODV protocols," *Advances in Science, Technology and Engineering Systems Journal*, vol. 5, no. 5, pp. 176–192, 2020.

[10] L. Qabajeh and M. Qabajeh, "Detailed performance evaluation of ARANz, ARAN and AODV protocols," *Journal of Theoretical and Applied Information Technology*, vol. 98, no. 12, pp. 2109–2131, 2020.

[11] L. Qabajeh, "A scalable secure routing protocol for managed-open mobile ad-hoc networks," *International Journal of Electrical and Electronic Engineering & Telecommunications*, vol. 11, no. 1, pp. 54–66, 2021.

[12] K. Sanzgiri, D. LaFlamme, B. Dahill, B. Levine, C. Shields, and E. Belding-Royer, "Authenticated routing for ad hoc networks," *IEEE Journal On selected areas in Communications*, vol. 23, no. 3, pp. 598–610, 2005.

[13] A. Hassani, A. Sahel, and A. Badri, "FTC-OF: Forwarding traffic consciousness objective function for RPL routing protocol," *International Journal of Electrical and Electronic Engineering and Telecommunications*, vol. 10, pp. 168–175, 2021.

[14] Z. Haas, M. Pearlman, and P. Samar, "The performance of query control schemes for the zone routing protocol," *ACM/IEEE Transactions on Networking*, vol. 9, no. 4, pp. 427–438, 2001.

[15] T. Lin, "Mobile ad-hoc network routing protocols: Methodologies and Applications," Ph.D thesis, Faculty of the Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 2004.

[16] M. Abolhasan, T. Wysocki, and E. Dutkiewicz, "A review of routing protocols for mobile ad hoc networks," *Ad Hoc Networks*, vol. 2, no. 1, pp. 1–22, Elsevier, 2004.

[17] Y. Cao and S. Xie, "A position based beaconless routing algorithm for mobile ad hoc networks," in *Proc. International Conference on Communications*, *Circuits and Systems*, vol. 1, pp. 303–307, IEEE, 2005.

[18] H. Li and M. Singhal, "A secure routing protocol for wireless ad hoc networks," in *Proc. 39th Annual Hawaii International Conference on System Sciences (HICSS '06)*, vol. 9, pp. 225a–225a, IEEE, 2006.

[19] S. Mizanur, M. Mambo, A. Inomata, and E. Okamoto, "An anonymous on- demand position-based routing in mobile ad hoc networks," in *Proc. International Symposium on Applications and the Internet*, pp. 300–306, Arizona, USA, 2006.

[20] A. Dorri, S. Kamel, and E. kheyrkhah, "Security challenges in mobile ad hoc networks: A survey," *International Journal of Computer Science and Engineering Survey (IJCSES)*, vol. 6, no. 1, pp. 15–29, 2015.

[21] S. Thapara and S. Sharmab, "Attacks and security issues of mobile ad hoc networks," in *Proc. International Conference on Sustainable Computing in Science, Technology & Management (SUSCOM)*, Jaipur, India, pp. 1463–1470, 2019.

[22] Z. Khan and A. Sharma, "Security Aspects of MANETs: A Review," *International Journal of Computer Science and Mobile Computing*, vol. 8, no. 7, pp. 40–44, 2019.

[23] F. Abdel-Fattah, K. Farhan, F. Tarawneh, and F. AlTamimi, "Security challenges and attacks in dynamic mobile ad hoc networks manets," *IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*, Amman, Jordan, pp. 28–33, 2019.

[24] M. Soni and B. Joshi, "Security assessment of routing protocols in mobile adhoc networks," in *Proc. International Conference on ICT in Business Industry and Government (ICTIBIG)*, IEEE, 2016.

[25] M. Soni and B. Joshi, "Security assessment of SAODV protocols in mobile ad hoc networks," *Data Science and Big Data Analytics*, vol. 16, pp. 347–355, 2018.

[26] J. Arshad and M. Azad, "Performance evaluation of secure on-demand routing protocol for Mobile ad hoc networks," *Sensor and Ad Hoc Communications and Networks Conference (SECON)*, IEEE, 2006.

[27] N. Luong, T. Vo, and D. Hoang, "FAPRP: A machine learning approach to flooding attacks prevention routing protocol in mobile ad hoc networks," *Wireless Communications and Mobile Computing*, 2019.

[28] M. Abu Zant and A. Yasin, "Avoiding and isolating flooding attack by enhancing AODV manet protocol (AIF_AODV)," *Security and Communication Networks*, 2019.

[29] M. Belgaum, Sh. Musa, M. Su'ud, M. Alam, S. Soomro, and Z. Alansari, "Secured approach towards reactive routing protocols using triple factor in mobile ad hoc networks," *Annals of Emerging Technologies in Computing (AETiC)*, vol. 3, no. 2, pp. 32–40, 2019.

[30] W. Alnumay, U. Ghosh, and P. Chatterjee, "A trust-based predictive model for mobile ad hoc network in internet of things," *Sensors*, vol. 19, no. 6, 2019.

[31] G. Borkar and A. Mahajan, "A review on propagation of secure data, prevention of attacks and routing in mobile ad-hoc networks," *International Journal of Communication Networks and Distributed Systems*, vol. 24, no. 1, pp. 23–57, 2020.

[32] M. Boulaiche, "Survey of secure routing protocols for wireless ad hoc networks," *Wireless Personal Communications*, vol. 114, pp. 483–517, 2020.

[33] V. Giruka and M. Singhal, "Angular routing protocol for mobile ad-hoc networks," in *Proc. 25th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW'05)*, pp. 551–557, 2005.

[34] L. Qabajeh, "A scalable multicast routing protocol for mobile ad-hoc networks," *Journal of Telecommunications and Information Technology*, vol. 3, no. 2, pp. 58–74, 2022.

[35] B. Karp and H. Kung, "GPSR: Greedy perimeter stateless routing for wireless networks," in *Proc. 6th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM 2000)*, pp. 243–254, Massachusetts, USA, 2000.

[36] X. Wu, "VPDS: virtual home region based distributed position service in mobile ad hoc networks," in *Proc. 25th IEEE International Conference on Distributed Computing Systems (ICSCS 2005)*, pp. 113–122, 2005.

[37] Y. Ko and N. Vaidya, "Location-aided routing (LAR) in mobile ad hoc networks," *Wireless Network (WINET)*, vol. 6, no. 4, pp. 307–321, ACM, 2000.

[38] L. Blazevic, L. Buttyan, S. Capkum, S. Giordano, J. Hubaux, and J. Le Boudec, "Self- organization in mobile ad-hoc networks: the approach of terminodes," *IEEE Communication Magazine*, vol. 39, no. 6, pp. 166–174, 2001.