

# Reinforcement learning-based security schema mitigating man-in-the-middle attacks in fog computing

Hossam Elmansy, Khaled Metwally, Khaled Badran

Department of Computer Engineering and Artificial Intelligent, Military Technical College, Cairo, Egypt

## Article Info

### Article history:

Received Sep 12, 2022

Revised Mar 13, 2023

Accepted Mar 28, 2023

### Keywords:

Fog computing security  
Internet of things security  
Man-in-the-middle  
Moving target defense  
Multi-path transmission control protocol  
Reinforcement learning  
Software defined networking

## ABSTRACT

The fast emerging of internet of things (IoTs) has introduced fog computing as an intermediate layer between end-users and the cloud datacenters. Fog computing layer characterized by its closeness to end users for service provisioning than the cloud. However, security challenges are still a big concern in fog and cloud computing paradigms as well. In fog computing, one of the most destructive attacks is man-in-the-middle (MitM). Moreover, MitM attacks are hard to be detected since they performed passively on the network level. This paper proposes a MitM mitigation scheme in fog computing architecture. The proposal mapped the fog layer on software-defined network (SDN) architecture. The proposal integrated multi-path transmission control protocol (MPTCP), moving target defense (MTD) technique, and reinforcement learning agent (RL) in one framework that contributed significantly to improving the fog layer resources utilization and security. The proposed schema hardens the network reconnaissance and discovery, thus improved the network security against MitM attack. The evaluation framework was tested using a simulation environment on mininet, with the utilization of MPTCP kernel and Ryu SDN controller. The experimental results shows that the proposed schema maintained the network resiliency, improves resource utilization without adding significant overheads compared to the traditional transmission control protocol (TCP).

*This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.*



## Corresponding Author:

Hossam Elmansy

Department of Computer Engineering and Artificial Intelligent, Military Technical College

37 Ismail Al Fangari, El-Qobba Bridge, Cairo, Egypt

Email: [hossamelmansy.developer@gmail.com](mailto:hossamelmansy.developer@gmail.com)

## 1. INTRODUCTION

Cloud computing has emerged as a significant trend in the field of information technology (IT) over the last twenty years. It encompasses various concepts, including the internet of content (IoC), internet of services (IoS), and internet of things (IoT), as part of its vision for the future development of IT, and supported by an expanding network infrastructure, where content, services, and things have become the main orientation of the new vision. The emergent growth in wearable devices, mobile devices, and sensors, have impacted end-user prospects, such that they are no longer satisfied with a traditional service provisioning paradigm. Custom quality of service (QoS) expectations for provisioned services are increasing and there is a rising expectation of the potential capability of cloud systems as an IT infrastructure to help create new value.

Recently, the IoT has gained significant attention and interest in recent years and is considered one of the most intriguing and rapidly growing areas in the current century. It converts trivial items into intelligent ones and enables communication and interaction among them [1]. This rapid development of IoTs has spawned many applications in numerous industries [2]. Many useful applications have been developed based on IoT include automotive, home automation, healthcare, and industry, among others [3]. In addition, communication

between IoT sensors/devices imposed new restrictions on the entire system's architecture, like privacy, traffic load, latency, and security [4]. Thus, fog computing and edge computing have been evolved to minimize the restrictions and limitations of cloud computing as well as to maintain the customers' high expectations. Fog computing is a novel computing paradigm that got considerable interest in recent years. It was firstly offered by Cisco in 2013 [5]. Figure 1 illustrates the fog computing paradigm layered hierarchy, the lowest layer represents the edge devices, e.g., mobile devices, sensors, wearable devices. The middle layer comprises fog nodes (network nodes) that provide storage, network connection, and computing functions. The top layer represents the cloud data center (CDC). Fog computing layer was primarily identified by its proximity to the edge devices and the processing capabilities it have that matched the restricted resources of the edge devices [6]. As a result, fog nodes is beneficial in time-sensitive services and real-time applications, in the sense that, fog nodes can perform analysis on data supplied by the edge devices (such as mobile devices) and then send back a real-time result to the edge devices [7].

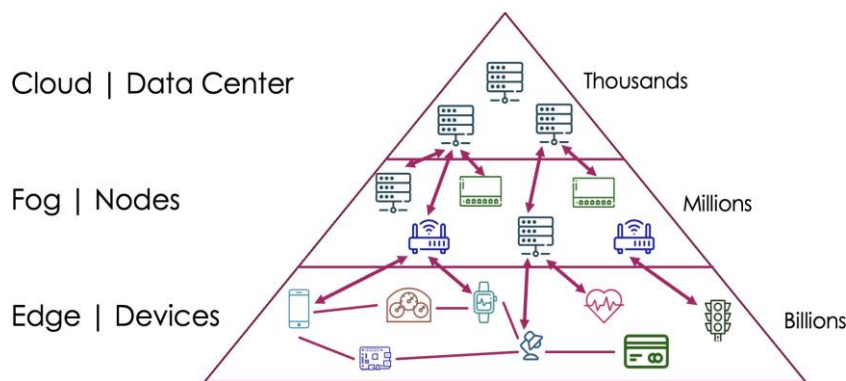


Figure 1. Fog computing hierarchy

The existence of direct connections between fog nodes had a significant impact on the data transmission efficiency in the fog layer network as well as between the edge devices [8]. Thus, it is urgently needed to implement flexible network traffic management that enhance the network efficiency and improve the network resources utilization in this layer. The successful implementation of software-defined network (SDN) in data centers prompted the usage of SDN in fog computing since they share similar goals of achieving both network resilience and high throughput [9]. SDN emerged as a promising technique compatible with fog computing networks for managing network traffic [10]. Moreover, SDN introduced additional capabilities, like its able to operate the network programmatically, management and centralized network control. SDN fundamental design divided both of the control plane and the data plane, allowing the SDN controller to automatically operate the whole network in a dynamic and adaptable manner. The SDN controller receives packets from the data plane in a continuous manner and sends the corresponding forwarding rules to the data plane. OpenFlow is a well-known protocol for SDN that enables remote management of network devices routing tables [11].

Since both SDN network topologies and fog computing share the same architecture characteristics, both are susceptible to the same risk. There are different types of attacks that can compromise SDN-based networks, including denial of service (DoS) and man-in-the-middle (MitM) attacks. These security threats can potentially disrupt the normal functioning of SDN networks and cause significant damage to the overall infrastructure [12]. It is worth mentioning that MitM attack is recognized as the most popular threat in fog computing [13], since the fog computing architecture is intrinsically similar to the MitM attack technique as shown in Figure 2, i.e., fog nodes reside between CDC's servers and edge devices [14]. Moreover, it is clear that fog nodes are closer to the attacker and have less computational power than cloud servers [15].

The static nature in traditional networks architecture makes attacks easily compromise the network and allows the attacker collect knowledge about the network states before delivering the attack effectively. Moreover, traditional intrusion detection systems (IDSs) and firewalls may be circumvented using malwares that are commonly available on the internet. Thus, moving target defense (MTD) [16] emerged as a novel security approach that tries to create asymmetric uncertainty on the attacker side. This increased the complexity and expense of conducting attacks and reduced the network vulnerability exposure as well. MTD approach seeks to change the target attack surface continually and randomly (by changing the flows routes, IP addresses, and port numbers).

Multipath routing [17], [18] is an existing technique that permits the simultaneous use of many routes in a network. It is worth mentioning that recent edge devices (e.g., mobile devices) had various network interfaces, including Wi-Fi and cellular that allowed the adoption of this multipath routing. Accordingly, multipath transmission control protocol (MPTCP) has been evolved as a modification to the traditional transmission control protocol (TCP) that enabled two hosts to interact simultaneously through several TCP connections created on various interfaces [19]. It is worth mentioning that recent applications and operating systems kernels have enabled MPTCP implementation. In addition, it works well in current existing networks, because of MPTCP backward compatibility, such as the internet, where regular TCP is used.

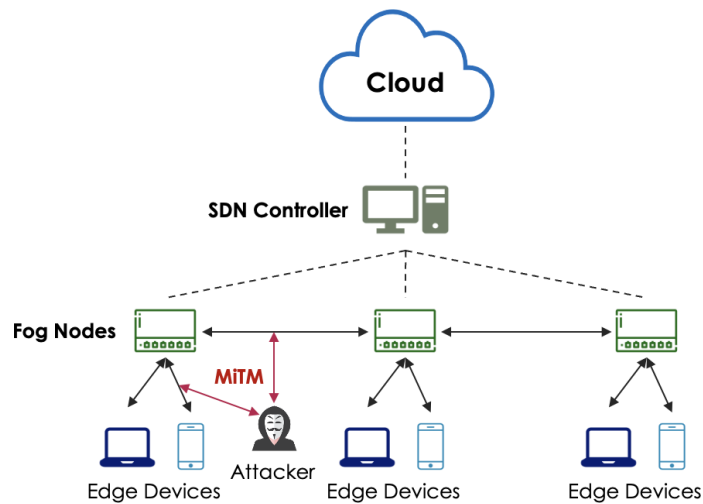


Figure 2. Fog computing threat model

In the last decades, numerous research studies have been conducted on fog computing implementation, particularly for IoT networks. The following related works highlighted some research proposals covering the adoption of SDN in fog computing and its associated security solutions. Li *et al.* [20], described MitM attacks against SDN networks that mainly depend on TLS protocol to protect the control channels between SDN controllers and SDN switches. In addition, they recommend a small countermeasure exploiting Bloom filters to identify this attack. However, if an OpenFlow connection between a switch and a controller in one path has been intercepted and an attacker updated fields that are not recorded in the Bloom Filter, their approach will not be effective.

Aliyu *et al.* [21] described a strategy for identifying and preventing intrusions, or MitM attacks. Each node in the IDS looked for fog nodes and calculated their arrival time to determine their response. MitM attacks, e.g., eavesdropping and packet alteration, were avoided by the intrusion prevention system's (IPSs) using lightweight encryption and decryption. However, their method was unsuccessful in a busy environment where packet delivery periods might differ widely.

Liu *et al.* [22] created a public-key cryptography-based cloud computing security framework; traditional public key infrastructure (PKI) based encryption. However, this approach not suited for fog computing [23] due to its high computational and communication overheads. To ensure the security of its nodes and data transmission, fog computing cannot rely solely on cryptographic techniques due to their demanding calculations and the limited resources available.

Chliyah *et al.* [24] presented a method for defending SDN networks against MitM attacks. In their solution, both SDN and MPTCP have been implemented. In addition, the MPTCP sub-flow routing was controlled by pathfinder and secure load sharing (SLS) algorithms. However, while their solution employed MPTCP to split traffic across different pathways, it did not provide automated network modifications to prevent network scans by attackers. In addition, if an attacker captured both MPTCP sub-flows, the whole communication could be successfully intercepted.

To enhance the mutation efficiency and increase the complexity of scanning and poisoning attacks, Zkik *et al.* [25] modeled SDN topologies, to determine acceptable routes automatically two new modules were built automatically using a pathfinder method. However, due to the deterministic nature of these multipath mutation algorithms, an attacker can determine the mutation path, placing at risk any packets transported along this path. In addition, not all accessible routes between source and destination were utilized.

Babar *et al.* [26] created an authentication system for IoTs that is resistant to eavesdropping and MitM attacks. Due to the limited computational power of IoT devices, it has been proposed to shift the required calculations to registration authority (RA) devices with higher computational capabilities. A suitable RA device in a fog computing environment is the fog node. However, if the attacker was able to abuse the fog node, the complete network is now vulnerable.

Several studies [27] have been done in the framework of protecting IoTs against MitM attacks. The most typical countermeasures for MitM risks were mutual authentication, encryption, and ensuring infected servers have been separated. As there are no relevant standard security standards for fog computing, these methods have not been modified for fog computing. In addition to authentication and encryption, other established security methods such as secure socket layer (SSL) and transport layer security (TLS) have been employed to safeguard data flow between fog nodes. Despite being one of the most widely used encryption systems, TLS still has flaws in both its cipher suites and the protocol.

MTD was developed in [28] to prevent inside and outside attacks on the SDN. In this case, the MTD method reduces threats by integrating MTD with the SDN environment and employing the hosts' virtual IP addresses. However, moving between multiple pathways causes a delay, and not all paths are being used.

This paper proposes a framework for mitigating MitM attacks in fog computing networks. As demonstrated in Figure 3, the proposed solution integrates SDN and MPTCP to harden the attack surface to MitM attacks. Moreover, the proposed framework incorporates MTD in two distinct ways. First, random host mutation (RHM) is accomplished by constantly switching the IP addresses of hosts. Second, random route mutation (RRM) [29] is achieved by constantly switching the routes between destination and source hosts. In addition, the suggested framework utilizes a reinforcement learning (RL) algorithm to determine the most efficient routes between source and destination hosts, hence minimize the latency and improve network throughput. This work's major objective is to avoid MitM attacks without incurring additional overheads on the network i.e., optimize the resource utilization. The rest of the paper is structured as follows: section 2 presents the proposed framework schema, including security considerations. Section 3 discusses the attack scenario, performance evaluation, and framework security schema implementation. Finally, section 4 concludes the paper.

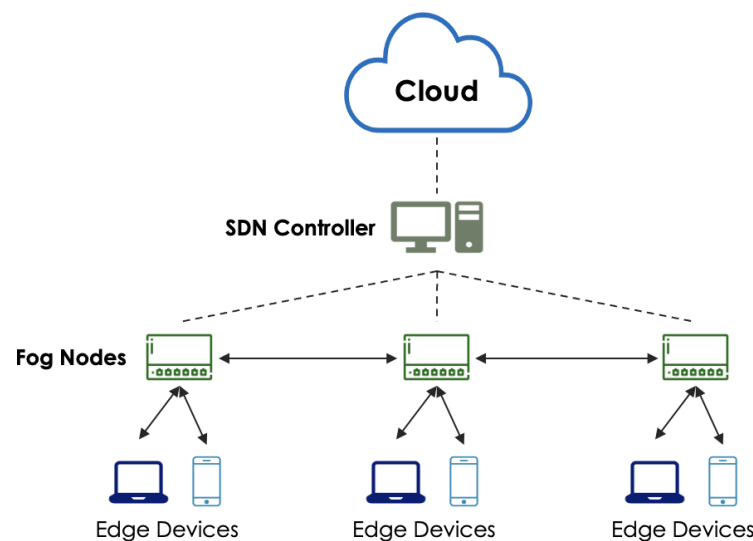


Figure 3. Fog-to-cloud architecture

## 2. METHOD

This study suggests a MitM reduction strategy that combines dispersed fog nodes in the fog computing layer with MPTCP, SDN, MTD, and RL-based routing agents. This deployment is made to handle various network configurations and apps. The working setting and the suggested remedy will be covered in more detail in the following subsections.

### 2.1. Working environment

The working environment used in the study is depicted in Figure 3, comprising three levels: “the cloud layer” as the first level, “the fog layer” as the second level, and “the edge devices layer” as the third level. The

edge devices, which include smartphones, IoT devices, sensors, and other gadgets, are present in the bottom layer. The data that is gathered and sent to the CDC servers by the fog nodes through the edge devices is thought to be of exceptional quality. The fog layer, which serves as a bridge between the cloud and edge devices, is the intermediate layer. Networking fog computing devices that handle data offloading, network connection, and fog computing services make up this layer. The CDC servers were represented by the top layer. A collection of edge devices covering a certain region and services may be offered by each fog node put at the network's edge.

The primary focus of this research is on the fog layer, which is the second layer of the working environment. Within the fog layer, the distributed architecture of fog nodes has made it possible to develop a distributed computing model. This computing model enabled the adoption of the SDN paradigm, MTD approach and applied RL-based routing algorithm in the network.

In the working scenario, the edge devices collect and upload data to the fog layer. Consequently, the fog layer performs analytics, classification capabilities, and data processing at the fog nodes (network's edge). In certain cases, the computations are transferred to the cloud layer for processing, while the results are transferred back to the third layer. Thus, fog nodes offer computational capabilities close to the edge devices, hence lowering end-to-end latency. Moreover, the cloud servers have the highest processing and computational capabilities that fog nodes and edge devices may require.

End-to-end service was intended to be provided via the proposed framework. Figure 4 shows the suggested framework applications' design. Fog nodes are used, which are positioned between edge devices and the cloud. A central SDN controller manages and regulates network traffic between the dispersed fog nodes, as well as between the fog nodes and edge devices. Fog nodes represented the scattered OpenFlow switches with limited processing power in the SDN network. These OpenFlow switches may provide specific limited services in addition to switching.

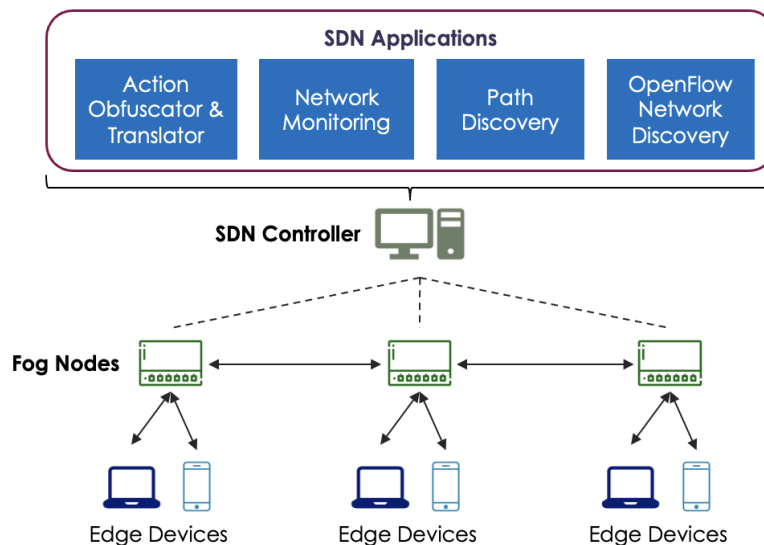


Figure 4. Proposed framework SDN-based applications

Using the OpenFlow protocol, the SDN controller manages all the installed OpenFlow switches (i.e., fog nodes) and orchestrates network traffic between them. Moreover, SDN-based applications that built up the proposed framework has been implemented on the top of the SDN controller. The proposed system uses of all available bandwidth and the redundant paths by employing MPTCP protocol to distribute traffic between fog nodes across various paths. It is worth noting that, MPTCP implementation ensured the network's resiliency against MitM attacks.

Figure 5 is a schematic representation of an attacker scenario in which the attacker attempted to conduct a MitM attack in the second layer. In addition, the attacker attempted to interrupt packets sent among fog nodes and edge devices. Since the proposed system employs MPTCP between the fog nodes and edge devices, and both host and route mutation have been used, it will be hard for an attacker to capture traffic. And even if the attacker succeeded in intercepting one of the sub-flows from the connection, it will be hard to capture the whole traffic because of the adoption of MPTCP.

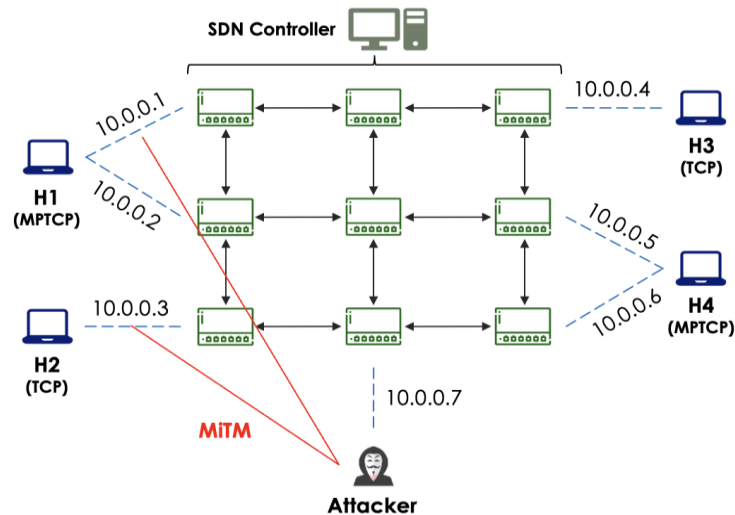


Figure 5. Fog nodes layer attack scenario

## 2.2. Proposed framework design

The proposed framework integrated SDN, MTD, MPTCP, and RL routing to mitigate MitM attacks in Fog network. Figure 4 illustrated the implemented SDN applications in a typical Fog architecture, as well as their interactions with other components. The SDN controller communicates with the Open vSwitches through protected OpenFlow channels. SDN applications are software modules that have been implemented atop the SDN controller and provide particular network operations. In proposed framework, the following modules has been implemented:

- OpenFlow network discovery application (ONDA): The network topology is discovered in this module using the link layer discovery protocol (LLDP) [30].
- Path discovery application (PDA): This module discovers all the available paths between source and destination nodes.
- Network monitoring application (NMA): This module collects the essential network information via passive and network measures. Network information includes capacity, throughput, link latency, and link status. In this proposal, the collected data represented the network states and the computed incentives.
- Action obfuscator and translator application (AOTA): This module is responsible for converting the actions generated by the RL agent into a series of OpenFlow messages, which are then used to update the routing tables of the Open vSwitches. The module periodically converts the actual IP addresses of the hosts to virtual IP addresses and also adjusts the MPTCP flow routes according to the data generated by the NMA application.

## 2.3. MPTCP

The growing number in the connected devices in modern networks, makes it difficult to anticipate the flowing traffic dependency since network traffic grows exponentially. Because of this, it may exist an MPTCP connection (two TCP sessions) that utilizes the same path, which can result in a successful MitM, if an attacker is able to intercept traffic along this way. For enhancing the network's efficiency, throughput, and latency, network traffic should be directed through the most optimal pathways. Therefore, RL-based routing algorithm has been implemented in SDN controller to find the optimal paths between source and destination nodes. RL agent will select the best  $n$  paths between the source and the destination based on throughput and latency metrics, then the best two paths will be selected for MPTCP connection.

## 2.4. MTD

The proposed architecture employed MTD to improve network unpredictability and prevent reconnaissance attacks, as well as prevent attackers from scanning the network and identifying the network's services. In this research, two distinct types of MTD have been implemented. The first technique used is RHM, which involves periodically modifying the IP addresses of nodes. With the help of the SDN controller, the actual IP address of each node is randomly replaced with a virtual IP address. Second, random route mutation (RRM), which continually altering the network traffic flow paths between the source and destination nodes. As MPTCP was the chosen protocol for communication, each MPTCP connection between two nodes will establish two TCP sessions. It is worth mentioning that, using RRM, RHM, and MPTCP harden the attacker trials

to conduct MitM attack. This will introduce a great deal of uncertainty into the network discovery. Adoption of MTD promoted security, complexity, and the expense of attacks while minimizing exposure to weaknesses.

## 2.5. Reinforcement learning routing

With the fast evolution of connected devices (smartphones, wearable devices, and IoTs) and network technology, network traffic increases tremendously. Therefore, it is challenging to understand and anticipate communication networks traffic, since they become increasingly complex and dynamic. Instead of constructing a precise mathematical model of the underlying network, intelligent agents were deployed in the network. The deployed agents represent RL agents that collect extensive network statistics like throughput, bandwidth, latency, and connection status.

In contrast to supervised learning and unsupervised learning, where the dataset contains the labels for the trained model, RL is constrained to learn from experience, i.e., learning from the collected data on the network status from the hosted environment. Thus, RL was selected, aiming to maximize the available network resources.

The conducted network in this research as shown in Figure 5 is represented by the directed graph  $G(N, \varepsilon)$ .  $N = \{node_1, node_2, \dots, node_n\}$  is the set of nodes in the network, i.e., 9 Open vSwitches, whereas  $\varepsilon$  is the set of links. It is assumed that the network connectivity is full duplex. A path  $p_{src,dest}$  is a graph walk that connects  $node_{src}$  and  $node_{dest}$  via a series of nodes. Also,  $bw_t(src, dest)$  represents the bandwidth of the link  $e_{src,dest}$  that connects  $node_{src}$  and  $node_{dest}$  at time interval  $\Delta_t$  and  $th_t(src, dest)$  represents the throughput of the link  $e_{src,dest}$  that connects  $node_{src}$  and  $node_{dest}$  at time interval  $\Delta_t$ . Table 1 provides a summary of the key notations used in this research.

Table 1. Notation definition

Notation	Definition
$N$	The set of nodes, $node_i : 1 \leq i \leq n$
$R$	The set of real IP addresses, $R_i : 1 \leq i \leq r$
$V$	The set of virtual IP addresses, $V_i : 1 \leq i \leq v$
$\varepsilon$	The set of links in the network
$e_{src,dest}$	The link that connects $node_{src}$ and $node_{dest}$
$\Delta_t$	The time interval
$node_{src}$	The source node
$node_{dest}$	The destination node
$p_{src,dest}$	The path that connects $node_{src}$ and $node_{dest}$
$bw_t(src, dest)$	The bandwidth of link $e_{src,des}$ at time $\Delta_t$
$th_t(src, dest)$	The throughput of link $e_{src,des}$ at time $\Delta_t$
$delay_t(src, dest)$	The delay of link $e_{src,des}$ at time $\Delta_t$
$status_t(src, dest)$	The link status of link $e_{src,des}$ at time $\Delta_t$
$f_1$	Bandwidth measurement for the $S_t$
$f_2$	Throughput measurement for the $S_t$
$f_3$	Latency measurement for the $S_t$
$f_4$	Connection Status measurement for the $S_t$

In a typical RL working environment, in which an agent that has been deployed on an Open Vswitch interacts with an environment  $E$  as shown in Figure 5 across discrete time intervals  $\Delta_t, t \geq 0$ . Each time interval  $\Delta_t$ , the agent watches the current state  $S_t$  and selects an action  $A_t \in A(S_t)$  where  $A(S_t)$  is the set of all available actions. In exchange, the agent obtains a reward  $R_{t+1}$  and advances to the subsequent state  $S_{t+1}$ . The procedure is repeated until the agent reaches a terminal condition. The agent's objective is to learn a policy  $\pi : S \rightarrow A$  that maximizes the expected future reward  $R = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$ , where  $\gamma \in [0,1]$  is the discounting factor. As actions use pathways, identifying the optimal policy is similar to determining the best routes in different network states. The first step of RL is to define the states, actions, and a scalar reward for the ultimate objective. The primary goal is to maximize throughput and minimize latency.

$$R = \sum_{t=0}^{\infty} \gamma^t R_{t+1} \quad (1)$$

First, a description for RL-routing will be introduced. Then, a Q learning technique will be provided to address the routing issue. The RL-routing model is depicted by  $M = \{S, A, R, \gamma\}$ , where:

- $S \in \mathbb{R}$  is the state space.
- $A$  is the action space.
- $R : S \times A \rightarrow \mathbb{R}$  is a reward function.
- $\gamma \in [0,1]$  is a discounting factor.

A state  $S$  at time interval  $\Delta_t$  is represented by (2):

$$S = [f_1, f_2, f_3, f_4] \quad (2)$$

At intervals of time  $\Delta_t$ ,  $S$  produces a summary of network information. The features are calculated in following manner.  $f_1 = \{bw_t(src, dest)\}$  is the link bandwidth at time interval  $\Delta_t$ .  $f_2 = \{th_t(src, dest)\}$  is the link throughput at time interval  $\Delta_t$ , where the throughput for a link  $e_{src,dest}$  is:

$$th_t(src, dest) = \frac{tx_t(e_{src,dest})}{bw_t(src,dest) \cdot |\Delta_t|} \quad (3)$$

where  $tx_t(e_{src,dest})$  is the amount of data that is transmitted via  $e_{src,dest}$  at time interval  $\Delta_t$ .  $|\Delta_t|$  is the duration of the time interval  $\Delta_t$ .  $f_3 = \{latency_t(src, dest)\}$  is the link latency at time interval  $\Delta_t$ .  $f_4 = \{status_t(src, dest)\}$  is the link status at time interval  $\Delta_t$ , where the status for the link  $e_{src,dest}$  is:

$$status_t(src, dest) = \begin{cases} 1 & \text{if } e_{src,dest} \text{ is up during } \Delta_t \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

The action space is represented as (5)

$$A = \{a_1, a_2, \dots, a_n\} \quad (5)$$

where an action,  $1 \leq i \leq n$ ,  $a_i(src, dest)$  is a collection of routes that connect  $node_{src}$  to  $node_{dest}$ . The controller's scalability issue has been taken into consideration in the action definition when determining the number of agents required for data forwarding. It gives the agent the ability to configure a one-to-many network all at once. Therefore, each Open vSwitch just requires only one agent deployed to it.

The reward function receives state  $s$  and an action  $a$  as inputs and produces a reward that represents the quality of the selected action. The reward function is defined as (6),

$$r = r_1 + r_2 \in [0,2] \quad (6)$$

where  $r_1$  and  $r_2$  are the selected action's  $a$  throughput and latency, respectively as (7) and (8).

$$r_1 = AVGIF(\sum_{p_{src,dest} \in a} \frac{tx_t(p_{src,dest})}{bw_t(p_{src,dest}) \cdot |\Delta_t|}) \quad (7)$$

$$r_2 = \sum_{p_{src,dest} \in a} delay_t(p_{src,dest}) \quad (8)$$

Overall, the agent has two main objectives which are reflected in the reward function detailed in (6). A higher reward indicates that a greater number of packets are successfully transferred between  $node_{src}$  to  $node_{dest}$  with minimal delay. This applies in both directions of data transmission.

To implement the RL routing algorithm, we used the dueling double deep Q-learning (dueling DDQN) architecture with prioritized experience replay [31] and the  $\epsilon$ -greedy policy to address the reinforcement learning problem. This design addresses the issue of inflated Q-values and improves the stability of learning.

Algorithm 1 represented the implemented module exploiting MTD and RL. In step 1, the RL agent realizes the network topology  $G(N, \epsilon)$  found by the OpenFlow network discovery application (ONDA). Step 2 involves using the path discovery application (PDA) to create an action space  $A$ , following (5), for the topology  $G(N, \epsilon)$ . This process generates a set of routes for all pairs of nodes in the topology. In steps 3 and 4, the SDN controller initializes the set of real and virtual IP addresses assigned to network nodes. The agent then initializes the network's current state, as in (2), by utilizing the NMA, the network current state is represented by the bandwidth, throughput, latency, and link status.

At every time interval  $\Delta_t$  (step 7), a random number is generated and used for RHM while the IP addresses of all network nodes are changed arbitrarily. In step 11, the agent chooses an action  $A_t$  that maximizes the reward  $R_t$ , as in (6). The agent executes  $A_t$  by calling the AOTA to update the Open vSwitches routing tables and change the IP addresses. It permits the network to operate for  $\Delta_t$ . The NMM is then invoked to collect updated network information  $S_{t+1}$ , as in (2). The collected network data used to compute the reward  $R_{t+1}$ , as in (6), for the selected action  $A_t$ . This process is repeated until the RL agent achieves the intended reward for final goal.



**Algorithm 1. Proposed system algorithm with MTD & RL**

```

1:  $G(N, \varepsilon) \leftarrow \text{OpenFlow Network Discovery}$ 
2:  $A = \text{PDA}(G(N, \varepsilon), \text{node}_{\text{src}}, \text{node}_{\text{dest}}, h)$ 
3:  $R \leftarrow \text{Initialize the set of virtual IP addresses}$ 
4:  $V \leftarrow \text{Initialize the set of real IP addresses}$ 
5: Initialize  $S = [f_1, f_2, f_3, f_4]$  by invoking NMA
6: repeat every  $t$ :
7:    $x \leftarrow \text{generate random number}$ 
8:   for each real IP address  $R_i$  in  $R$ :
9:     Map each  $R_i$  to  $V_i$ 
10:  end for
11: choose Random action  $A$ , or  $A_t = \text{argmax } q_\pi(S_t, A)$ 
12: execute  $A_t$  by invoking AOTA
13:  $S_{t+1} = [f_1, f_2, f_3, f_4]$  by invoking NMA
14: compute the reward  $R_{t+1}$  for action  $A_t$ 
15: end repeat

```

### 3. RESULTS AND DISCUSSION

To evaluate the suggested framework and assess its performance, simulations were conducted on a virtual machine that featured an Intel(R) Xeon(R) CPU E5-2676 v3 @2.40 GHz processor and 4.00 GB of memory. The virtual machine was running the Linux Ubuntu 18.04 LTS operating system. We built the fog nodes layer network, or SDN network, in mininet, which included nine Open vSwitches and five hosts, as shown in Figure 5. For the hosts used for the tests, the MPTCP Linux kernel implementation was installed. For the fog nodes layer, we used the Ryu SDN controller. Both H1 and H4 hosts had two network interfaces and utilized the MPTCP protocol. The three hosts, Host-2 (H2), Host-3 (H3), and Host-5 (H5) utilized the TCP protocol and shared a single network interface. Attacker Host-5 (H5) uses Ettercap to launch a MitM attack against Host-1 (H1) (H2). The default speed for each open vSwitch port is 10 Gb. The network's performance was evaluated using iPerf in four distinct conditions: standard TCP traffic flow, TCP traffic flow with a MitM attack simulated using an RL agent developed in Python, MPTCP traffic flow, and MPTCP traffic flow with a MitM attack also simulated using the RL agent.

RL agent has been trained to determine the optimal network paths connecting two parties. By setting  $h = 8$ , PDA has been executed to generate an action space with eight paths between a source and a destination. Then, the best two paths have been selected for the sub-flows to establish the source-to-destination MPTCP connection. Source-destination host combination has been created to generate random traffic for RL agent training purposes. Additionally, the data transmission patterns between hosts are produced randomly.

Different values for the duration of a time interval  $|\Delta_t|$  have been tested. It was observed that when  $|\Delta_t| = 1\text{s}$ , for all metrics, RL-routing offers the best performance. The agent has been trained in an episodic fashion. Each episode consists of 100 steps, where the duration of each step is  $|\Delta_t|$  seconds.

The proposal was tested with various traffic compositions, including traffic that is created randomly, for training purpose. It was discovered that orchestrating hosts to provide the same amount of traffic throughout each episode is an efficient method of training the agent. This technique forced the agent to return to each stage frequently enough to test out various options. We plan for each host pair to create traffic on a regular basis in a set of predetermined steps. We allowed each pair of hosts to randomly generate traffic with a duration obtained from a normal distribution with  $\mu = 5$  seconds and  $\sigma = 1$ . Data transfer patterns between each host pair are left up to the generator. The duration of the burst and the packet interval time are not specified.

Figure 6 depicts the agent's training while using RL-routing. The figure illustrated the performance of the RL agent in terms of total rewards, using a window size of 50 episodes. On average, 25 k distinct states were produced, some of which are quite similar and others which are completely different. Figure 6 illustrates the number of training episodes on the x-axis and the total incentives earned during each episode on the y-axis. Initially, the RL agent does not comprehend the underlying network well enough. Thus, the agent largely engaged in environmental exploration and reaps little benefit. Following a few episodes, the awards rise until they reach their maximum levels.

The bottom layers of the dueling network in the chosen dueling DDQN are convolutional, just like in the original DQNs (3 convolutional layers followed by 2 fully connected layers). Two sequences (or streams) of fully connected layers have been employed in this architecture as opposed to one series of fully connected layers after the convolutional layers. Because of how the streams are built, different estimates of the value and advantage functions were produced. According to [31], the output of the network is a set of Q values, with one for each action, and the two streams are merged to create a unified output Q function. It is worth mentioning that the adopted dueling DDQN architecture and hyper-parameters as mentioned in [32], [33] have been fine tuned to fit the network state input vector that maintained the RL-routing operation.

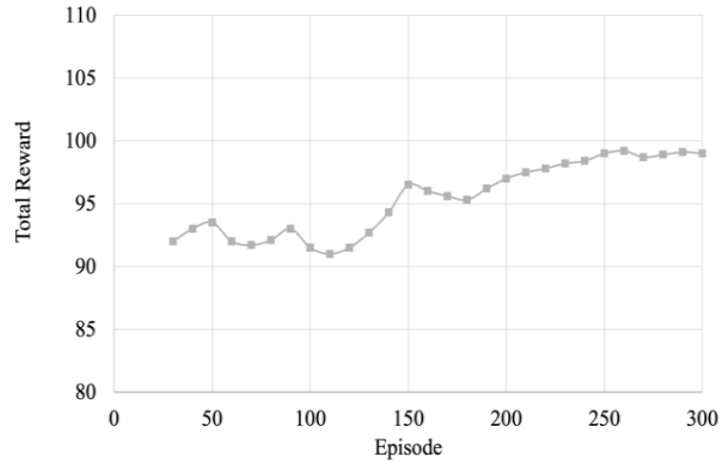


Figure 6. RL agent performance

The objective is to understand what and how to discover an effective solution to the routing issue given the state representation, reward function, and action description. It is worth mentioning that any routing algorithm uses a policy to determine its route without losing generality. A stochastic policy  $\pi$  determines the routing algorithm's behavior and is represented as a distribution over actions for a specific state, defined (9).

$$\pi(a|s) = P[a|s] \quad (9)$$

For an agent behaving according to a stochastic policy  $\pi$ , the values of the state-action pair  $(s, a)$  and the state  $s$  (i.e.,  $Q, V$  functions) are defined as (10).

$$\begin{aligned} Q^\pi(s, a) &= \mathbb{E}[R_t | s_t = s, a_t = a, \pi], \text{ and} \\ V^\pi(s) &= \mathbb{E}_{a \sim \pi(s)}[Q^\pi(s, a)] \end{aligned} \quad (10)$$

The preceding  $Q$  function value can be computed recursively with dynamic programming:

$$Q^\pi(s, a) = \mathbb{E}_s \left[ r + \mathbb{E}_{a' \sim \pi(s')} [Q^\pi(s', a')] \right] | s, a, \pi \quad (11)$$

We define the optimal  $Q^*(s, a) = \max_\pi Q^\pi(s, a)$ .

Under the deterministic policy  $a = \arg \max_{a \in A} Q^*(s, a)$ , it follows that  $V^*(s) = \max_a Q^*(s, a)$ . From this, it also follows that the optimal  $Q$  function satisfies the Bellman (12)

$$Q^*(s, a) = \mathbb{E}_s \left[ r + \gamma \max_{a'} Q^*(s', a') \right] | s, a \quad (12)$$

We define another important quantity, the *advantage function*, relating the *value* and  $Q$  functions as (13).

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s) \quad (13)$$

Note that  $\mathbb{E}_{a \sim \pi(s)} [A^\pi(s, a)] = 0$ . The *value function*  $V$  intuitively measures how favorable it is to be in a particular state  $s$ , while the  $Q$  function measures the value of selecting a particular action when in that state. The *advantage function* is obtained by subtracting the state value from the  $Q$  function, providing a relative measure of the significance of each action.

RL-Routing algorithm begins with an initial policy  $\pi_{RL}$  and utilizes policy iteration to enhance its performance. In each step, the agent updates  $Q_{\pi_{RL}}$  by computing states using (2), selecting actions using its policy  $\pi_{RL}$ , and calculating rewards using (6). By repeating these steps for a large number of iterations, the agent identifies an optimal  $Q_{\pi_{RL}^*}$ . Once an optimal policy  $\pi^*$  is obtained, the agent selects the best possible action in each state  $s \in S$  by:

$$A = \arg \max Q_{\pi^*}(s, a), \quad a \in A(s) \quad (14)$$

Finding the optimal policy  $\pi^*$  is therefore comparable to finding the best effective solution (i.e., best routing path). RL-routing has the potential to discover an improved policy, as it depends on the traffic patterns observed during the training phase and the chosen exploration mechanism that manage the trade-off between exploration and exploitation. At the start of the training phase, the agent largely explores to gather information, i.e., exploration, as depicted in Figure 6. After some time, it begins to gain by leveraging its understanding of the underlying network to inform better decisions, i.e., exploitation.

Network throughput and delay overhead have been chosen as the assessment measures to measure how well the suggested architecture performs. Figures 7 and 8 compare the delay overhead between TCP and MPTCP with and without the MitM attack, while Figures 9 and 10 compare TCP and MPTCP in the presence and absence of MitM attacks. The tests were run repeatedly, and the average of the results was calculated.

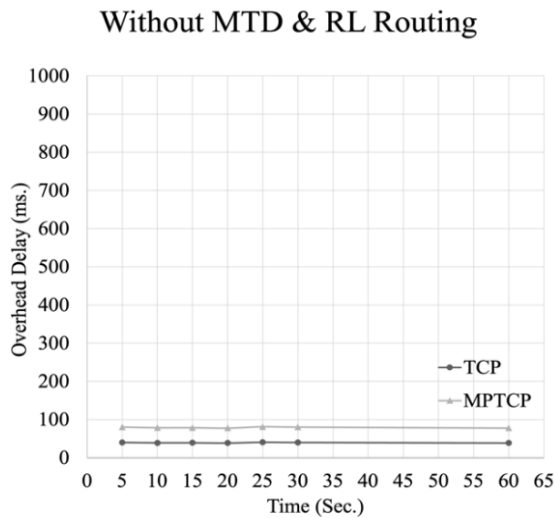


Figure 7. Delay overhead comparison for TCP vs. MPTCP

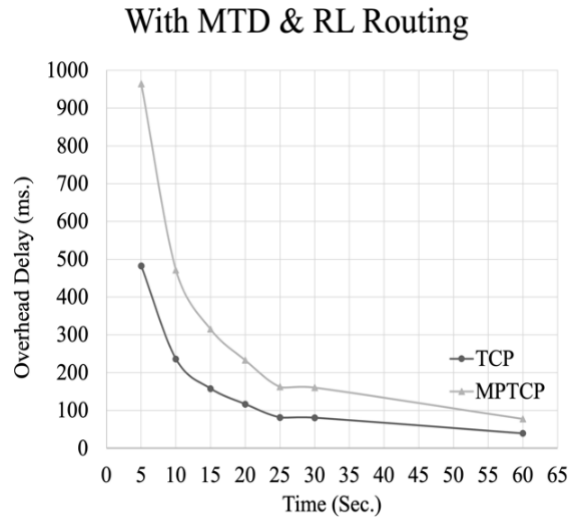


Figure 8. Delay overhead comparison for TCP vs. MPTCP

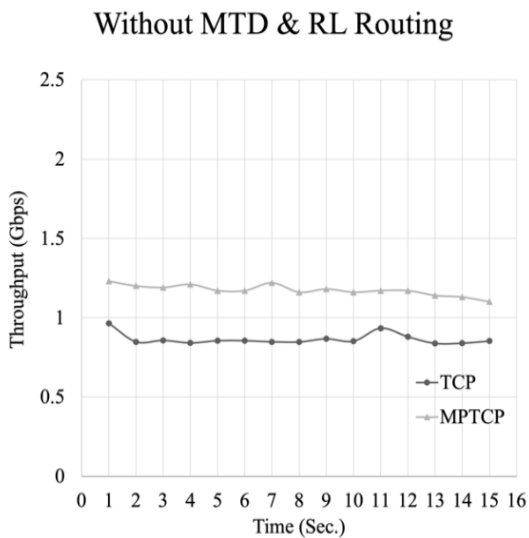


Figure 9. Throughput comparison for TCP vs. MPTCP

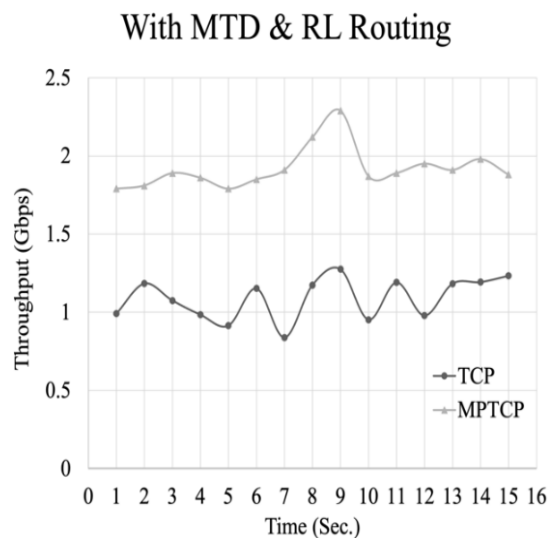


Figure 10. Throughput comparison for TCP vs. MPTCP

Figure 7 showed the overhead delay comparison between TCP and MPTCP without using MTD and RL. It showed that the delay overhead added by MPTCP is almost double of TCP as it uses two TCP sub flows for each MPTCP connection. Additionally, Figure 8 showed the overhead delay comparison between TCP and

MPTCP using MTD and RL routing with different configurations for the changing (mutation-turn around) time between new routes and IP addresses. It showed that when decreasing the changing time, the more delay overhead occurred. It is worth mentioning that, the degradation in performance ensure that it will be hard to the attacker to discover the network and reconnaissance it, it is a tradeoff between security and performance. Moreover, it showed that the proposed framework did not add much delay overhead to the network if the changing period configured to be 60 seconds.

The performance difference between TCP and MPTCP with/without MTD and RL routing was shown in Figures 9 and 10. The results collected demonstrated that the MPTCP network throughput was higher than that of conventional TCP. This is as a result of MPTCP using various paths between hosts. Also, the network throughput has been increased using our proposed framework as MPTCP sub flows since RL routing selected the best routes between source and destination. Exploiting RL routing helped in finding the best routes between two parties in terms of throughput and delay. This guaranteed MPTCP's involvement in preserving the fog nodes layer network's resilience and end-users' quality of hope (QoE).

A comparative study has been carried out among similar related work. Table 2 summarized this comparative study in terms of the defense methodology and the evaluation metrics. It was shown that, the proposed system has better security as it incorporates several techniques to mitigate MitM attacks in fog computing without adding a lot of overhead and delay to the network. As well as the overhead added by the complexity of the security solution have been controlled by using RL agent. Adoption of RL agent in routing optimization and selecting the best routing paths between nodes decreased the additional overheads incurred by the security solution.

Table 2. Comparison against related work

Authors/Publication	Defense methodology	Metrics
Aliyu <i>et al.</i> [21]	SDN, MITM	Time cost to detect attack and delay
Liu <i>et al.</i> [22]	IDS, IPS, MITM	Time cost to detect attack and delay and Energy Consumption
Zkik <i>et al.</i> [25]	SDN, MPTCP, MITM	Application execution time
Babar <i>et al.</i> [26]	SDN, MTD, MITM	---
Duan <i>et al.</i> [29]	SDN, MTD, MITM	---
Proposed Solution	SDN, MPTCP, MTD, Reinforcement learning, MITM	Throughput and delay

#### 4. CONCLUSION

This study proposed a MitM mitigation solution for fog computing that integrated SDN, MTD, MPTCP, and RL routing. The SDN controller has been configured for managing and controlling fog nodes. In addition, MPTCP has been implemented to make use of the various connection interfaces in fog computing Open vSwitches and edge devices, which enabled redundancy path for traffic between all fog nodes and edge devices. The system employed MTD in two ways, RHM and RRM, to enhance network uncertainty and harden the network scanning for attackers, hence decreasing the likelihood of a successful MitM attack. In addition, a RL routing algorithm has been employed to assist the SDN in determining the optimal network path between source and destination nodes. The simulation findings demonstrated that the proposed framework improved network throughput, robustness, and security with minimal delay overhead. In the future, we plan to enhance the security of the fog layer by leveraging additional SDN and MPTCP capabilities.




#### REFERENCES

- [1] J. Iannacci, "Internet of things (IoT); internet of everything (IoE); tactile internet; 5G – A (not so evanescent) unifying vision empowered by EH-MEMS (energy harvesting MEMS) and RF-MEMS (radio frequency MEMS)," *Sensors and Actuators A: Physical*, vol. 272, pp. 187–198, Apr. 2018, doi: 10.1016/j.sna.2018.01.038.
- [2] P. Sethi and S. R. Sarangi, "Internet of things: Architectures, protocols, and applications," *Journal of Electrical and Computer Engineering*, vol. 2017, pp. 1–25, 2017, doi: 10.1155/2017/9324035.
- [3] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: a survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015, doi: 10.1109/COMST.2015.2444095.
- [4] P. P. Ray, "A survey on internet of things architectures," *Journal of King Saud University - Computer and Information Sciences*, vol. 30, no. 3, pp. 291–319, Jul. 2018, doi: 10.1016/j.jksuci.2016.10.003.
- [5] N. L. Fantana *et al.*, *Internet of things - converging technologies for smart environments and integrated ecosystems*, River Publishers, 2013.
- [6] E. G. M. Petrakis, S. Sotiriadis, T. Soultanopoulos, P. T. Renta, R. Buyya, and N. Bessis, "Internet of things as a service (iTaaS): challenges and solutions for management of sensor data on the cloud and the fog," *Internet of Things*, vol. 3–4, pp. 156–174, Oct. 2018, doi: 10.1016/j.iot.2018.09.009.
- [7] X. Lyu, C. Ren, W. Ni, H. Tian, and R. P. Liu, "Cooperative computing anytime, anywhere: Ubiquitous fog services," *IEEE Wireless Communications*, vol. 27, no. 1, pp. 162–169, Feb. 2020, doi: 10.1109/MWC.001.1900044.




- [8] K. Pyatkova *et al.*, “Flood impacts on road transportation using microscopic traffic modelling techniques,” in *Simulating Urban Traffic Scenarios*, 2019, pp. 115–126, doi: 10.1007/978-3-319-33616-9\_8.
- [9] C. Lin, G. Han, X. Qi, M. Guizani, and L. Shu, “A Distributed mobile fog computing scheme for mobile delay-sensitive applications in SDN-enabled vehicular networks,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5481–5493, May 2020, doi: 10.1109/TVT.2020.2980934.
- [10] A. Mayoral, R. Vilalta, R. Muñoz, R. Casellas, and R. Martínez, “SDN orchestration architectures and their integration with cloud computing applications,” *Optical Switching and Networking*, vol. 26, pp. 2–13, Nov. 2017, doi: 10.1016/j.osn.2015.09.007.
- [11] P. Parol and M. Pawlowski, “Towards networks of the future: SDN paradigm introduction to PON networking for business applications,” in *2013 Federated Conference on Computer Science and Information Systems*, 2013, pp. 829–836.
- [12] Y. Desmedt, “Man-in-the-middle attack,” in *Encyclopedia of Cryptography and Security*, Springer US, 2005, pp. 368–368, doi: 10.1007/0-387-23483-7\_241.
- [13] B. N. B. Ekanayake, M. N. Halgamuge, and A. Syed, “Review: security and privacy issues of fog computing for the internet of things (IoT),” in *Cognitive Computing for Big Data Systems Over IoT*, 2018, pp. 139–174, doi: 10.1007/978-3-319-70688-7\_7.
- [14] B. Potter and B. Fleck, *802.11 security, ser. O’Reilly series*. O’Reilly Media, Incorporated, 2003.
- [15] S. Khan, S. Parkinson, and Y. Qin, “Fog computing security: a review of current applications and security solutions,” *Journal of Cloud Computing*, vol. 6, no. 1, Dec. 2017, doi: 10.1186/s13677-017-0090-3.
- [16] A. Aydeger, N. Saputro, and K. Akkaya, “A moving target defense and network forensics framework for ISP networks using SDN and NFV,” *Future Generation Computer Systems*, vol. 94, pp. 496–509, May 2019, doi: 10.1016/j.future.2018.11.045.
- [17] R. Banner and A. Orda, “Multipath routing algorithms for congestion minimization,” in *NETWORKING 2005: NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems*, 2005, pp. 536–548, doi: 10.1007/11422778\_43.
- [18] M. Chhiah, G. Orhanou, and S. El Hajji, “Countering MitM attacks using evolved PathFinder algorithm,” *International Journal of Cloud Applications and Computing*, vol. 7, no. 2, pp. 41–61, Apr. 2017, doi: 10.4018/IJCAC.2017040104.
- [19] C. Paasch and O. Bonaventure, “Multipath TCP,” *Queue*, vol. 12, no. 2, pp. 40–51, Feb. 2014, doi: 10.1145/2578508.2591369.
- [20] C. Li, Z. Qin, E. Novak, and Q. Li, “Securing SDN infrastructure of IoT–fog networks from MitM attacks,” *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1156–1164, Oct. 2017, doi: 10.1109/JIOT.2017.2685596.
- [21] F. Aliyu, T. Sheltami, and E. M. Shakshuki, “A detection and prevention technique for man in the middle attack in fog computing,” *Procedia Computer Science*, vol. 141, pp. 24–31, 2018, doi: 10.1016/j.procs.2018.10.125.
- [22] J. Liu, Y. Xiao, and C. L. P. Chen, “Authentication and access control in the internet of things,” in *2012 32nd International Conference on Distributed Computing Systems Workshops*, Jun. 2012, pp. 588–592, doi: 10.1109/ICDCSW.2012.23.
- [23] H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, M. Aledhari, and H. Karimipour, “A survey on internet of things security: Requirements, challenges, and solutions,” *Internet of Things*, vol. 14, Jun. 2021, doi: 10.1016/j.iot.2019.100129.
- [24] M. Chhiah, G. Orhanou, and S. El Hajji, “SDN MPTCP sub-flows routing security against MitM attacks,” *International Journal of Control and Automation 1*, vol. 11, no. 6, pp. 123–136, 2018, doi: 10.14257/ijca.2018.11.6.12.
- [25] K. Zkik, A. Sebbar, Y. Baddi, and M. Boulmalf, “Secure multipath mutation SMPM in moving target defense based on SDN,” *Procedia Computer Science*, vol. 151, pp. 977–984, 2019, doi: 10.1016/j.procs.2019.04.137.
- [26] S. Babar, A. Stango, N. Prasad, J. Sen, and R. Prasad, “Proposed embedded security framework for internet of things (IoT),” in *2011 2nd International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*, Feb. 2011, pp. 1–5, doi: 10.1109/WIRELESSVITAE.2011.5940923.
- [27] Y. Sheffer, R. Holz, and P. Saint-Andre, “Summarizing known attacks on transport layer security (TLS) and datagram TLS (DTLS),” Feb. 2015, doi: 10.17487/rfc7457.
- [28] S. Macwan and C.-H. Lung, “Investigation of moving target defense technique to prevent poisoning attacks in SDN,” in *2019 IEEE World Congress on Services (SERVICES)*, Jul. 2019, pp. 178–183, doi: 10.1109/SERVICES.2019.00050.
- [29] Q. Duan, E. Al-Shaer, and H. Jafarian, “Efficient random route mutation considering flow and network constraints,” in *2013 IEEE Conference on Communications and Network Security (CNS)*, Oct. 2013, pp. 260–268, doi: 10.1109/CNS.2013.6682715.
- [30] “IEEE Standard for local and metropolitan area networks - station and media access control connectivity discovery,” in *IEEE Std 802.1AB-2016 (Revision of IEEE Std 802.1AB-2009)*, 2016, pp. 1–146, doi: 10.1109/IEEESTD.2016.7433915.
- [31] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, “Dueling network architectures for deep reinforcement learning,” *Prepr. arXiv.1511.06581*, Nov. 2015.
- [32] H. van Hasselt, A. Guez, and D. Silver, “Deep reinforcement learning with double Q-learning,” *Prepr. arXiv.1509.06461*, Sep. 2015.
- [33] V. Mnih *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

## BIOGRAPHIES OF AUTHORS






**Hossam Elmansy**    received the B.Eng. degree in computer engineering from Military Technical College, Cairo, Egypt in 2012. Currently, he is a Solutions Architect. His research interests include cloud computing, network security, multi-cloud, blockchain, AWS cloud, fog computing, IoT, databases, distributed computing, and security. He can be contacted at email: [hossamelmansy.developer@gmail.com](mailto:hossamelmansy.developer@gmail.com).



**Khaled Metwally**    received the B.Sc. and M.Sc. degrees in electrical engineering from Military Technical College (MTC), Cairo, Egypt, in 2001 and 2008, respectively, and the Ph.D. degree in electrical engineering from University of Ottawa, Canada, in 2017. He is currently a researcher and lecturer in Computers Eng. & Artificial Intelligence department in MTC. His research interests include cloud computing resources allocation management, provisioning, and optimization, cloud security, software-defined network (SDN) security, the applications of machine learning and deep learning techniques in object detection, robot navigation, cyber security techniques. He can be contacted at email: k.metwally@mtc.edu.eg.



**Khaled Badran**    received a Bachelor Degree in computer engineering and Masters of Science degree from the MTC, Cairo, Egypt, in 1995 and 2000, respectively. He also received the Ph.D. degree in Electrical and Computer engineering from Sheffield University, UK, in 2009. He is currently Head of the Department of Computer Engineering and Artificial Intelligence, MTC. His research interests are in artificial intelligent, data mining, semantic web, and database security. He can be contacted at email: khaledbadran@mtc.edu.eg.