Republic of Iraq
Ministry of Higher Education and Scientific Research
University of Babylon
College of Information Technology
Department of Information Networks

PROPOSED CRYPTOGRAPHY KEY MANAGEMENT IN SDN ENVIRONMENT

A Thesis

Submitted to the Council of the College of Information Technology at University of Babylon in Partial Fulfillment of the Requirements for the Degree of Master in Information Technology / Information Networks.

By

Huda Haider Mohammed AL-Hamdany

Supervised by

Prof.Dr. Wesam Sameer AbdAli Bhaya

2023 A.D **1445 A.H**

يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنْكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ وَاللَّهُ بِمَا تَعْمَلُونَ خَبِير

Supervisor Certification

I certify that the thesis entitled (**PROPOSED CRYPTOGRAPHY KEY MANAGEMENT IN SDN ENVIRONMENT**) was prepared under my supervision at the department of Information Networks/ College of Information Technology / University of Babylon as partial fulfillment of the requirements of the degree of Master in Information Technology-Information Networks.

Signature:

Supervisor Name: **Prof. Dr. Wesam S. Bhaya**

Date: / /2023

The Head of the Department Certification

In view of the available recommendations, I forward the thesis entitled "**PROPOSED CRYPTOGRAPHY KEY MANAGEMENT IN SDN ENVIRONMENT"** for debate by the examination committee.

Signature:

Prof. Dr. **Alharith A.Abdullah**

Head of Information Networks Department

Date: / /2023

Declaration

I hereby declare that this thesis, submitted to the University of Babylon in partial fulfillment of requirement for the degree of Master of Information Technology-Information Networks has not been submitted as an exercise for a similar degree at any other university. I also certify that this work described here is entirely my own except for experts and summaries whose sources are appropriately cited in the references.

Signature:

Name : Huda Haider Mohammed AL-Hamdany

Date: **11 / 10 / 2023**

Dedication

I dedicate this thesis

To my father

To my mother

To my husband

To my supervisor

To my family

To my friends

Researcher

Acknowledgements

In the name of God, Most Gracious, Most Merciful, At first, Praise be to God and thanks to God and the satisfaction of parents and conciliation only from God greatest praise is to **Allah** for His assistance in facing the difficulty that I met in my study, and for always helping me to achieve my aims, also for His great graces and boons all the time.

I would like to express my deepest thanks to my supervisor **Prof. Dr. Wesam S. Bhaya** for their valuable advice, motivation, guidance, and for so many fruitful discussions throughout the preparation of this thesis.

I would like to extend my respect and deepest gratitude to the College of Information Technology.

Sincere appreciation and love go to my family, my father and my mother are always be with me whatever they provide me with optimism and pure affection and they give me great hope, encouragement and they have stood with me in every step in this research. I dedicate this work and give special thanks to those who encouraged me to continue my scientific career, my husband Abdulla and beautiful son Ali.

Finally, Sincere thanks and appreciation to all friends, colleagues and loved ones

Researcher

Abstract

Software-defined networking (SDN) is one of the best technologies that allows network management through software by separating routing operations (control plane) from the data plane and making the network control process through one central part, which is the controller. The control plane consists of one or more controllers and acts as the brain of the network. This separation led to many benefits, including cost reduction, programmability, and ease of network management.

The separation added a problem related to the security in many parts of the network, especially in the data plane, because the messages are not encrypted, so any attack can eavesdrop on the messages and gain access to the sensitive data and mitigate the security requirement.

This thesis proposes a method to encrypt data at the data plane and dynamically manage encryption keys in an SDN environment by embedding them inside certificates and distributing those keys over the Secure Socket Layer (SSL) to all trusted devices in the data plane. The Rivest–Shamir–Adleman (RSA) encryption algorithm was relied upon to achieve confidentiality of data transmitted over the network.

As a result, the data will become encrypted, and the network will be more secure so that attackers will not be able to cause any network data exposure. On the other hand, as a result of using dynamic coding management, the network performance does not deteriorate. The best results for the average throughput stability of the proposed secure SDN system is 12.366 Gbps and the average time is 8.111 seconds of packet flow.

List of Contents

List of Tables

Table of Figures

List of Algorithms

List of Abbreviations

Abbreviation	Description
API	application programming interface
ARP	Address Resolution Protocol
CA	Certification Authority
DHCP	Dynamic Host Configuration Protocol
DOS attacks	Denial of Service attacks
FIFO	First-In First-Out
IOT	Internet-of-Things
KDC	Key Distribution Center
KTC	Key Translation Center
M.L	Machine Learning
MAC	Media Access Control
MITM Attack	Man In The Middle Attack
PKI	Public Key Infrastructure
RAs	Registration Authorities
RSA	Rivest–Shamir–Adleman
SDNs	Software-defined networks
SSL	Secure Socket Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Protocol
TLV	Type Length Value
VBNs	virtual networks

List of Thesis Related Publications

Name of Conference: The Sixth International Iraqi Conference on Engineering Technology and its Applications (6th IICETA 2023).

- **1st Paper Title: A Proposed Cryptography Key Management in Software-Defined Networking (SDN).**

Authors:

- Huda Haider Mohammed AL-Hamdany
- Prof.Dr.Wesam S.Bhaya

Information Network Department, Information Technology College, Babylon University.

The Islamic University

51
11/April /2023

LETTER OF ACCEPTANCE

(A Proposed Cryptography Key Management in Software-Defined Networking (SDN))

Huda AL-Hamdani and Wesam S. Bhaya

It has been accepted for presentation in the **"The Sixth International Iraqi Conference on Engineering Technology and its Applications (6th IICETA 2023)"**. The final decision of publication in IEEE explore is subject terms and conditions of Conference Scientific Committee and IEEE.

Sincerely,

Conference Organizing Committee - COC 2023

The Sixth International Iraqi Conference on Engineering Technology and its Applications (6th IICETA 2023)

Chapter One

General Introduction

1.1 Introduction

The key aspects of any secure communications network are authentication, non-repudiation, information availability, confidentiality, and integrity. To protect a network from malicious attacks or unintentional damage, security specialists must protect data, the hardware of the network, and the communication activities over the network. Security is a problem that is becoming increasingly important, particularly in software define networks(SDNs) , and encryption can be used at many points throughout the network to improve security[1].

The SDNs are a relatively new model that introduces the concept of abstraction, which means separating the control plane from the data plane and making decision-making and control centralised by the controller, who has a comprehensive understanding of the entire network, while the hardware is only responsible for forwarding or passing packets to their destination according to the instructions of the controller, Which are usually a set of rules for handling packages [2].

The term abstraction in SDN means that the network device has been separated into two separate levels: the control level and the data level, and therefore each level is dealt with separately, so that algorithms and protocols in the control level can be developed easily without the need to make any change in the hardware infrastructure. The data level in the network contains devices only, so the role of this layer is only to pass data [3].

The SDN structure has been divided into three layers: the first is the application layer, which consists of the services and applications provided by the network to the user; the second is the control layer, which contains the controller, who is the basis of this technology and is responsible for routing decisions, management, and control of all network functions; and the third layer is the infrastructure layer, or called the data layer, which consists of network devices, where these devices receive orders from the controller layer and execute them [4].

In the network structure, the control plane needs to communicate with network devices in the data plane, where these plane are located at the bottom of the control plane. Therefore, the controller must use an interface to communicate between them, and due to its location in the network the interface has become known as the southbound interface, as the Openflow protocol is one of the most famous technologies used as the interface. And the controller also needs to communicate with the application plane through the application programming interface (API), which is called the northbound interface [5].

The SDN networks added many advantages, the most important of which is the programmability, the ability to program the level of control for the network devices more precisely by changing the logic in which they operate. The ability to manage the network from a single location is due to the other benefit of having a global perspective on the network [6].

Despite the network characteristics that contributed to increasing the effectiveness and performance of the network, in contrast to these advantages, SDN networks have led to the introduction of new challenges in

terms of security; for example, the separation between the control level and the data level has increased the exposure of the network to denial of service (DOS) attacks [7].

1.2 Related works

Various relevant studies have been compiled and synthesized under the umbrella of related works.

A reliable architecture is proposed in (Yigit et al., 2019) to avoid problems related to data exchange in SDN networks. An application was proposed to configure encryption keys to create certificates to secure communications within the network. In this paper, the transport layer security (TLS) security protocol was used between nodes to provide security, integrity, and confidentiality, with the presence and use of the certificate ensuring the reliability of the connected parties. Security can be enhanced by implementing access control in order to reduce the risk of private key theft [8].

Security techniques for detection of attacks in distributed SDN infrastructure (Kallol et al., 2019) . Authorization Policy-based Security Architecture (APbSA) is proposed which enables specification of enforceable access policy constraints on communications and flows between end users/devices and services in SDNs across multiple domains. The APbSA is a trusted component of the security architecture and forms part of the SDN Controller. Another significant component of the security architecture is the security component in the SDN switches. It developed a security-enhanced OpenFlow switch with security component that can monitor the state of the switch and validate the flow rules as well as protect

the flow traffic for confidentiality and integrity using encryption mechanisms [9].

A novel Three-Tier investigated in (Ali et al., 2020) for intrusion detection and prevention system in software defined network. Not all the authentication users can be legitimate, since they are compromised, so that the major contribution is to identify all the compromised devices by knee analysis of the packets. Routers are the edge devices employed in first tier which is responsible to validate the IoT user with RFID tag and encrypted signature. Then the authenticated user's packets are submitted into second tier with switches that validates the packets. Then the key features are extracted from packets and they are classified into normal, suspicious and malicious. Then suspicious queue packets are classified and predicted using deep learning method. The proposed work is experimented in OMNeT++ environment and the performances are evaluated in terms of intruder Detection Rate, Failure Rate, Delay, Throughput and Traffic Load [10].

A secure application management framework is proposed by (Hu et al., 2021) based on REST API access control. It is to granularly manage application permissions and encrypt REST API calls to defend against malicious attacks. Secure application management framework based on REST API access control in SDN. Besides, SEAPP is a lightweight logic architecture between application plane and control plane and supports quick deployment and reconfiguration in runtime. Both theoretical analysis and evaluation results show the security and effectiveness of SEAPP. Besides, SEAPP introduces negligible CPU and memory overheads [11].

In (Lowery et al., 2021) developed solutions to classify encrypted OpenFlow traffic into OpenFlow message types. It examines the

effectiveness of two traffic classification techniques using a dataset generated from a simulated SDN, and shows that the techniques can achieve an accuracy up to 95%. The most successful features used to classify encrypted OpenFlow messages are explained along with a methodology of collecting data, labeling data, identifying features, and the training of models to achieve high accuracy of classification [12].

The problem of weak security is proposed in (Ghaly et al., 2021) as the process of transmitting data sent over the network by presenting and implementing several encryption algorithms and analysing the results of each algorithm and its strength in protecting that data. Where the AES algorithm and the RSA algorithm were applied separately, the two algorithms were also combined, and the results were monitored. The results showed that the RSA algorithm is almost equivalent to its encryption strength if combined, whereas when combined with the AES algorithm, it gives high encryption results. However, in return, it leads to a slowdown in data transmission time by applying it to several cases in the network [13].

The contemporary methods for SDN is proposed by (Jafarianet al., 2021) to find anomalies and mitigation are organized, compared, and discussed. They divided the SDN anomaly detection methods into five categories: flow calculating, information-based, entropy-based, hybrid, and deep learning. The current research gaps and major issues with SDN anomaly recognition are presented[14].

A study on Man-in-the-middle attack (MITM) is used in (Morsy al., 2022), which is a type of attack that causes the problem of breaching data confidentiality through an untrusted third party accessing, reading, and modifying data. Where the attacker associates his address with the address

of another legitimate host, this paper presents a mechanism to detect the presence of MITM without altering the ARP structure. To establish a connection between requests and responses, send an ARP packet with a trusted key along with the original packet. These links are stored in a specific file, and then this file is compared to see if there is a duplicate MAC address. To add more reliability, a DHCP server is used to discover the MAC addresses of attackers. This method is considered effective in terms of detecting the presence of plagiarism, but at the same time, it led to a slowdown in the speed of data transfer [15]. Table 1.1 showed the problem, methodology and results of the related works.

Table 1.1: Related Works summary.

Ref. No, Year	Methodology	Problem	Results
[8], 2019	Transport Layer Security (TLS), Access Control List (ACL)	Security architecture to Address security problems regarding data exchange in software- defined networks	Ensuring the reliability of the connected parties.
[9], 2019	Authorization Policy-based Security Architecture (APbSA)	Lack of trust between different devices and users	The results show that there is an increase in the latency with the APbSA compared to the baseline (without the APbSA). Also note that the latency increases linearly with the increase in the number of switches with the APbSA.
[10], 2019	User validation, packet validation and flow validation	The use of self-defined policies, multi-queues and flow verification that were not effective in detecting attacks	The increase in detection ratio eventually decreases the failure ratio, The failure rate decreases with the increase in switches; this is due to the sharing of packets that are received from users.
[11], 2021	Secure Application Management Framework (SEAPP)	Securely management the applications in the SDN-enabled	Evaluation results show the effectiveness of SEAPP to secure data in SDN
[12],2021	Transport Layer Security (TLS)	Outsider threat has gain access to information sent across networks	TLS results in lack of data visibility to network monitors, it, can prevent timely detection of and response to various network events
[13], 2021	advanced encryption	separating the control unit	The results showed that the hybrid

	standard (AES), Rivest–Shamir–Adleman (RSA), and hybrid encryption algorithms	from the data unit led to a problem related to poor security of data sent in the network	coding method is better in terms of security and improved time (faster than RSA alone)
[14], 2021	flow counting scheme, information-based scheme, entropy-based scheme, deep learning, and hybrid scheme	The various security attacks that the SDN network is exposed to, such as anomalies, intrusions, denial of service (DoS) attacks, etc. On the other hand, the significant impact of these attacks on the entire system and network.	techniques and methods used for collecting statistical data, different algorithms were used for detecting an anomaly. analysis results reported in different studies revealed that collecting data by OpenFlow protocol in networks leads to the saturation of the control plane. Hence, the use of specific protocols is highly essential.
[15], 2022.	Detection scheme with MAC protocol	A Man-in-the-middle attack (MITM) which an unauthorized third party secretly accesses the communication between two hosts in the same network to read/modify the transferred data between them.	It is considered a complete solution to detect all forms of ARP spoofing and identify the attacker without any changes in ARP protocol. which installed on all hosts.

1.3 Problem Statement

The use of SDNs in many important applications has led to many advantages. Many methods and techniques suggest improving the network's performance in order to benefit from SDNs in the communication process. But the problem that remained unresolved was the issue of security. When sending data from the sender to the recipient, this data is not encrypted and can be seen by any other party, so in this study the proposal is contributes to solving the problem of encrypting communications over the network using the RSA algorithm, but the other problem is how to ensure the delivery of keys for encryption to users securely and the management of those keys without affecting the effectiveness and performance of the network. Due to the extremely high dynamics in SDN networks, key distribution cannot be

achieved manually, so it was necessary to propose a scheme that achieves the distribution of encryption keys automatically and securely.

1.4 Aims of the thesis

The aim of this thesis is to create a key management scheme to control the distribution of asymmetric keys of RSA algorithms for all trusted devices in the data layer so that no untrusted device can receive those keys and later use those keys within the RSA cipher equation to encrypt all data and then send it to the other end to re-decrypt the message and read it.

The objectives of this thesis are as follows:

- Designing a scheme for key management.
- Managing distribution of asymmetric keys only to the authorized device.
- Verifying user certificates and ensuring data encryption by RSA algorithm.

1.5 Thesis Outline

There are five chapters in the thesis. Each chapter starts with a short introduction that says what the chapter is about and what its main points are. The following is a summary of each chapter:

- **Chapter One** : It gives a general overview of the research area. It shows the problems with this study and emphasises how important it is.
- **Chapter Two**: It provides the background details necessary to comprehend the research area that the thesis is based on. Central technologies such as RSA certification, Open vSwitch, and RYU are then also explained.
- **Chapter Three**: It explains all the equipment, algorithms, and methods that were used in the SDN environment to implement the proposal.

- **Chapter Four** : It contains the results of the proposed system and its evaluation.
- **Chapter Five** : It provides the results, conclusions, and future works.

Chapter Two

Theoretical Background

2.1 Introduction

This chapter defines the Software Defined Network (SDN) in detail and its planes, covering each plane's fundamental components and how to establish communication between planes. It also describes network security, cryptography, keys, and algorithms. Finally, briefly summarize the tools and techniques used mostly for experimentation and as a foundation when creating schema.

2.2 Software Defined Network (SDN)

The general concept of SDNs is to separate the control plane from the data plane and consolidate network management and decision-making under the controller, the network's brain that has a comprehensive view of the network. This abstraction induced by the SDN network makes network management easier , requires less equipment and more flexibility [16].

Figure 2.1 shows the general concept of the SDN network, where the controller communicates with every device (switches and routers) in the data plane via the OpenFlow protocol. In addition to the application plane that is located above the control level, which has the ability to programmed the network directly in real time via an API interface [17].

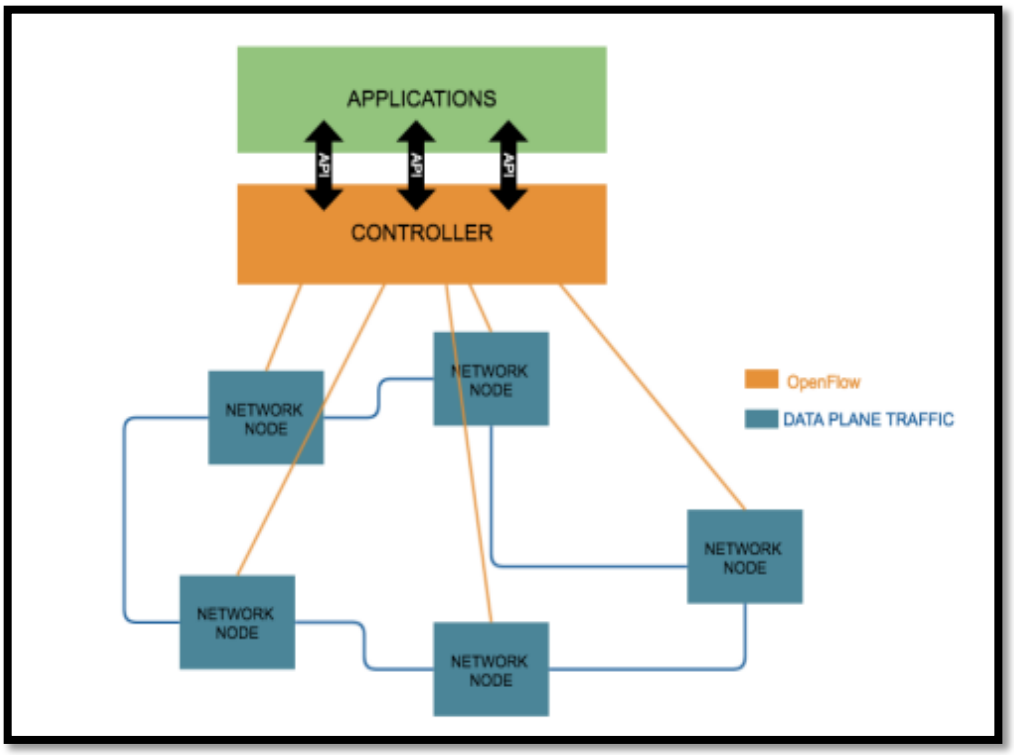


Figure 2.1: SDN Network[17].

Previously, in traditional networks, any device consisted of three levels: management, control, and data. Briefly, the function of the data level is to pass and direct the data. In other words, this layer works with data only.

The control level refers to any event that controls the data level, for example, creating tables by certain algorithms. The management level contains protocols that allow network operators to monitor, control, and manage devices on the network. Thus, modifying the logic of devices in the presence of those layers is impossible because they act as black boxes [18].

The device manufacturer traditionally builds and integrates the control and data planes into the same device, as shown in Figure 2.2, which operates according to a "hardware-centric" architecture. The result significantly increased the network's performance and resiliency. However, it is challenging to set up, troubleshoot, and administer this architecture [19].

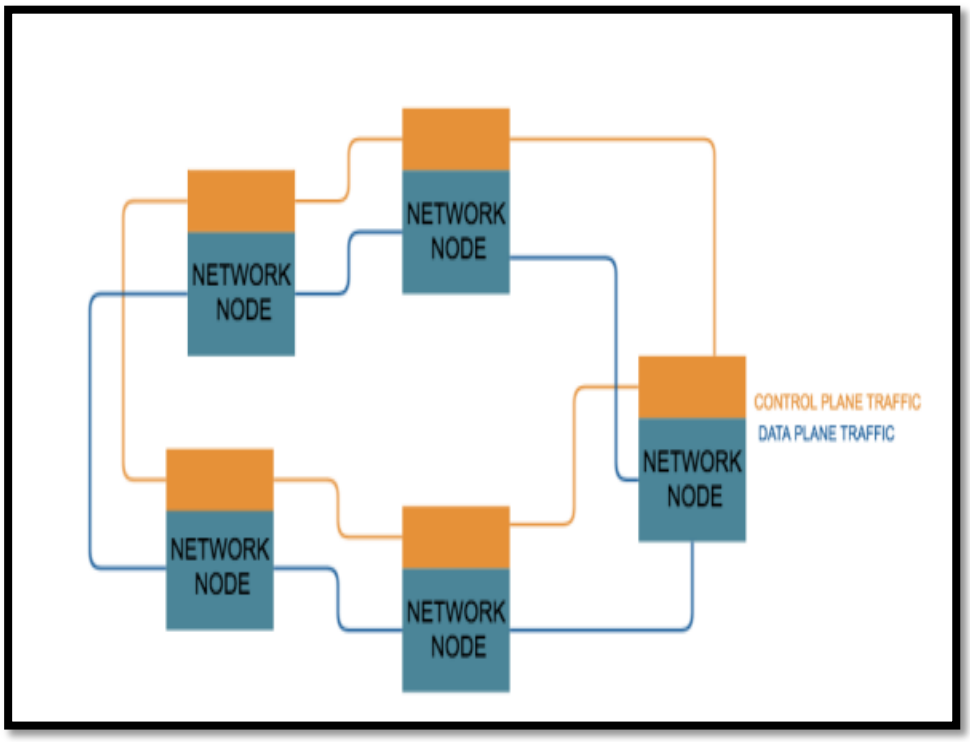


Figure 2.2: The Device In Traditional Network

After the 2009 release of the OpenFlow protocol, which will be explained later, the SDN architecture began to take shape. Many benefits such as cost savings, openness and other advantages have made many trends and interest towards the SDN network[20].

2.2.1 Architecture and Main Components of SDN

The SDNs framework makes networks programmable also enables the ability to autonomously govern and regulate network behavior using open interfaces. Keeping a centralised view of each part of the data flow throughout the network [21].The three basic levels that comprise the architecture are depicted in Figure 2.3 as the Data Plane, Control Plane, and Application plane [22]. In the next sections, it provided a brief overview of each of these aspects.

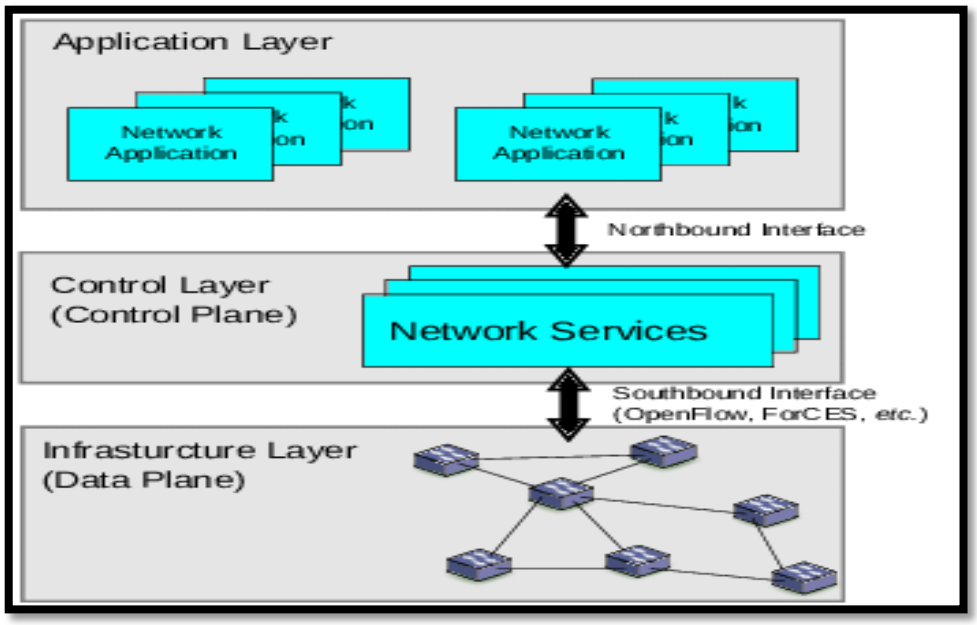


Figure 2.3: Main Layers of SDN[22].

1. Data Plane

The ability of the data plane is restricted to transferring data from one point to another and from one system to another using commands and rules that are managed by the controller, so it is sometimes called the forwarding plane [23].

The data plane includes devices, routers, and switches. It can also include virtual networks (VBNs) and even specialised networks, such as the Internet of Things(IOT) [24]. These planes interact with the controller using common OpenFlow interfaces, guaranteeing device interoperability and configuration compatibility [25].

2. Southbound API of SDN

The southbound interface is an important component of the SDN system that acts as a link between the data plane and the control plane. It allows the controller to manage the flow tables in the switches, thereby

controlling the network behaviour. A wide variety of southbound APIs, including OpenFlow, POF, OpFlex, and OpenStack. The most widely used southbound standard at the moment is OpenFlow [26].

3. SDN Controller

The "brain" of the network's and the main task is to create configurations for networks based on established rules. It offers lower-level details to the application plane through required services and popular APIs for programmers. There are many different types of controllers, and each one has a different design and architecture. Whether a controller has a distributed or centralised design [27].

Single point of failure issues and scaling restrictions are the two key drawbacks of centralised controller. NOX-MT, Maestro, Beacon, RYU, and Floodlight are the most popular controllers of this kind that can provide the throughput required for the networks [28].

Distributed controllers do this by scattering independent controllers over diverse network segments like ONOS, Onix, HyperFlow, PANE, and DISCO. It provides significantly more scalable support and is more resistant to physical and logical failures. [29] . With the help of eastbound and Westbound APIs, distributed controllers can communicate with one another and exchange data, consistency model algorithms, and monitoring data. Despite this, it suffers from bigger problems, the most important of which is the complexity and low productivity of the network, as well as affecting the speed of network performance. [30].

In a networked context, the first four components are principles to offer features like separation and security between services and applications[30].

4. Northbound API Of SDN

One of the most essential SDN abstractions is the northbound API. To connect controllers with apps, the northbound API offers mostly an interface. The low-level instruction needed to configure the forwarding hardware from southbound interfaces is abstracted. In fact, a number of controllers, including Floodlight, Onix, and OpenDaylight, build their northbound APIs utilising a variety of standards and programming languages. Because the Application Plane can contain a wide variety of apps, there is currently no industry-standard API for such activities as network virtualization techniques, cloud computing system control, security apps, and other divergent or specialised activities [30].

5. Application Plane

The applications control the actions of the devices. The control plane is provided with high-level policies by the applications and implements these policies as flow rules on devices to execute them. More than one type of application (including security, visualisation, and other applications) makes up an SDN application plane [31].

Numerous SDN apps have already been created; however, the focus is on developing SDN-compatible software that enables users to download and run network apps automatically. Most SDN applications fall under one of these five types: wireless and mobility, measurement, tracking,

security, and traffic engineering. There is a group of apps that make use of SDN's features to provide new kinds of solutions [32].

2.2.2 OpenFlow

OpenFlow is the most widely used type of protocol for sending messages between the controller and switches, where OpenFlow switches implement the protocol and through it allows the controller to take control and manage flow table rules as showing in Figure 2.4 [33].

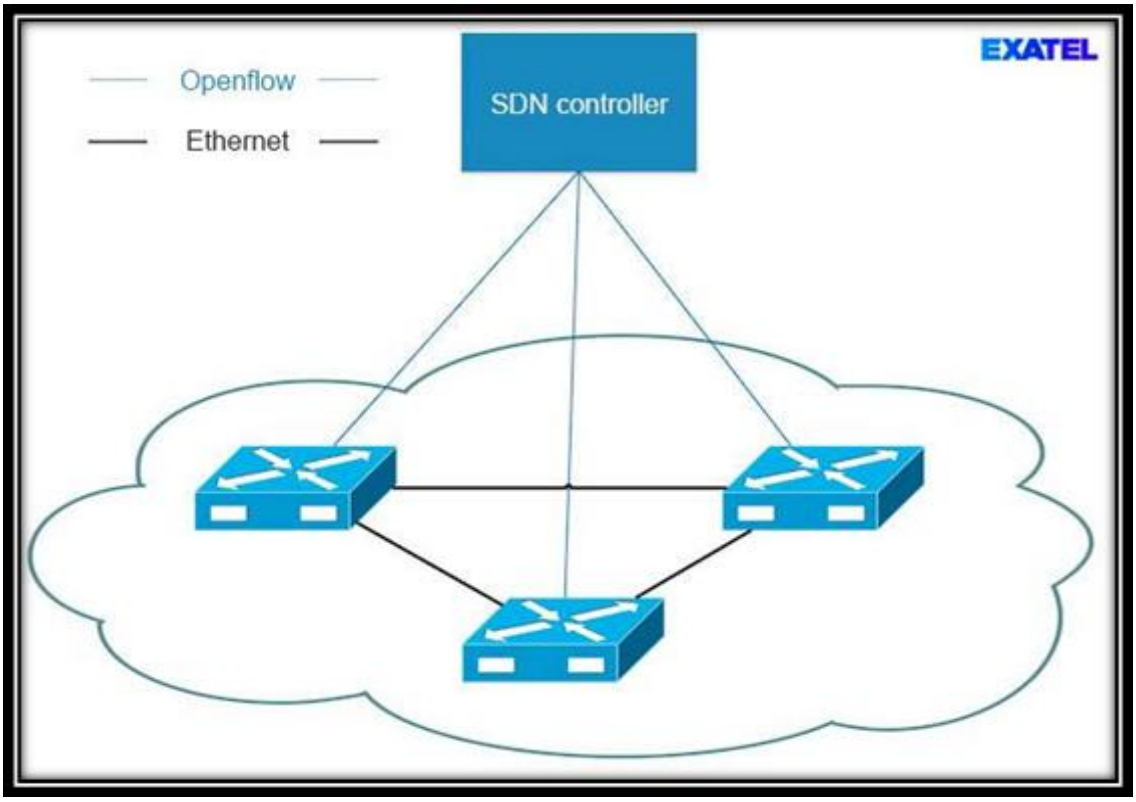


Figure 2.4: Openflow of SDN controller[33].

The OpenFlow protocol sends messages using Transmission Control Protocol (TCP) to guarantee reliable message delivery. A Type Length Value (TLV), which can fit 64KB of data inside one message, might

be used to give more variable values within a packet in OpenFlow communications [34]. As shown in Figure 2.5, the protocol has three different types of messages: asynchronous, controller-to-switches, and symmetric[35]:

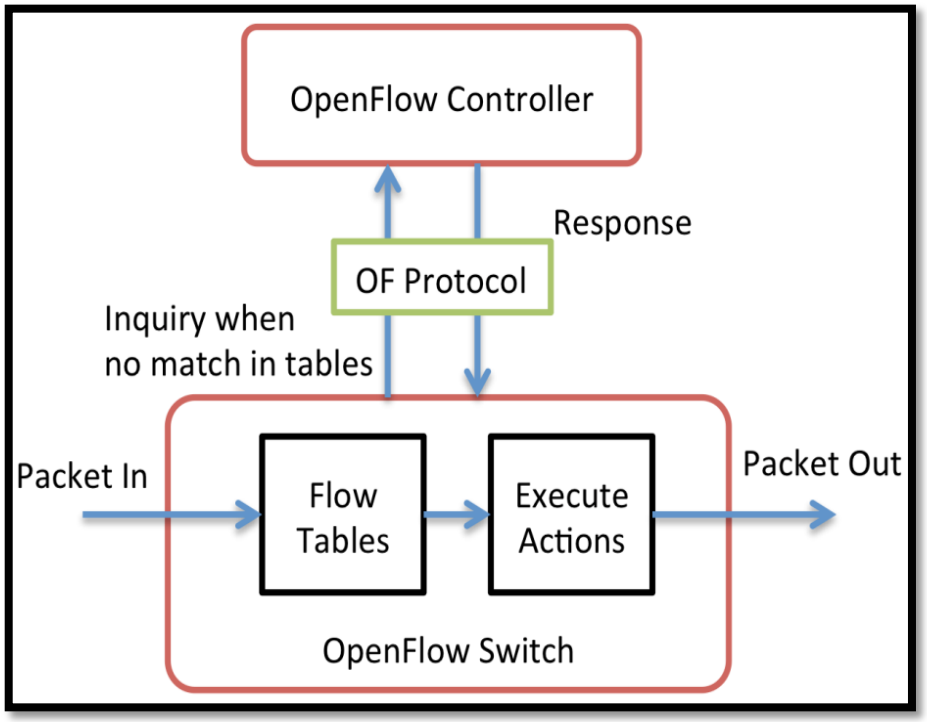


Figure 2.5: Three Different Messages of OpenFlow[35].

Asynchronous messages such as the packet-In message are sent to the controller when packets of data arrive and do not match any field in the flow table, or in the case of an action field that contains a procedure "send to the controller" The second message is controller-to-switches, such as Packet-Out; its function is to send the required data to the switches, and it is a response to a Packet-In message, which is a message that contains a specific command in order to add it to the flow table. And the last one is symmetric, which is typically used to set up or test a system [35].

The Figure 2.6 depicts the main components of an OpenFlow switch. The secure Channel is one of the key components of an OpenFlow switch that protects the transmission process between the controller and the switches. The secure channel supplies asymmetrical encryption based on TLS, but it is not essential [36].

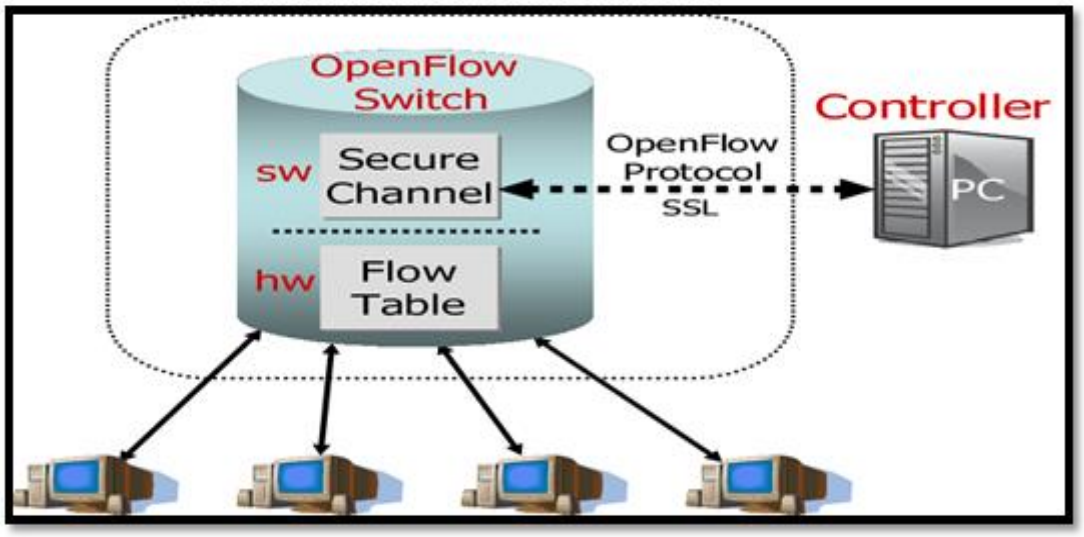


Figure 2.6: Components Of OpenFlow in SDN [36]

The other component is the flow table. The processing of incoming packets at the switches depends on a number of flow tables. The flow tables include many entries for specifying the flow depicted in Figure 2.7, and the packets will be forwarded by this entry as indicated by the first field of a flow table entry, called the match field. This can be demonstrated by matching flows that have the same IP, packet type, header, or any other field. The "table-miss" entry determines what to do about the packets if no entries match. The priority field separates packets that match multiple items. Counters are used to keep track of network statistics. Additionally, the packet's processing is determined by the instruction field. This field consist of a series of operations, such as dropping a packet or sending it to the controller [37].

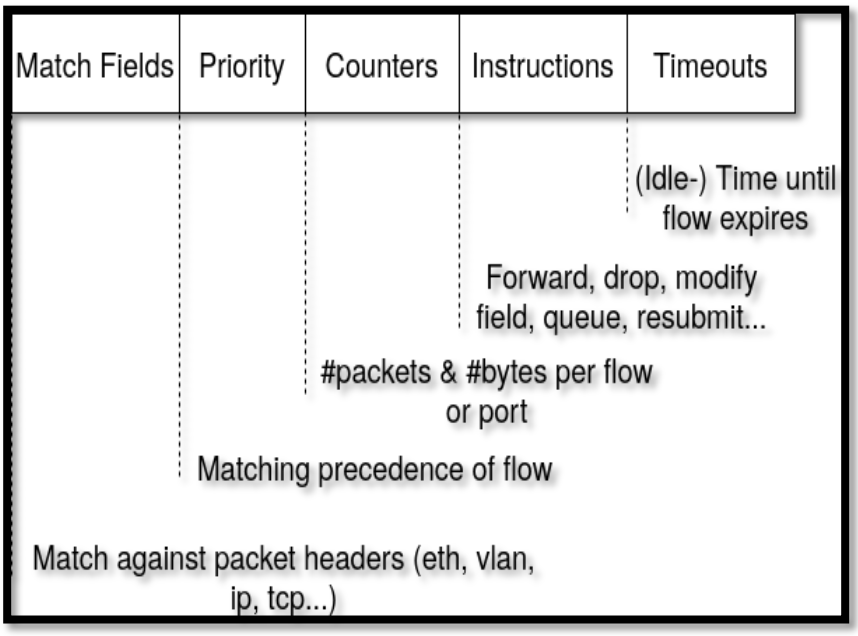


Figure 2.7: Flow table entries in OpenFlow switch[37]

The proactive and reactive modes of inserting packets into switches are available to a controller that uses the OpenFlow protocol. Reactively means that if the packets do not fit into any entry in the flow tables, they will be routed to the controller, and the controller will choose what will be done to the packet in this situation. The other mode is proactive mode, which means the controller will send the rules before commencing, and if packets that do not fit any values in the entry are dropped or assigned as "table miss" entries, this entries did not inform how to handle the packet that did not match all other table flow entries. The default action field is send all packets to the controller [38].

2.2.3 Security in SDN

SDN, like any other technology, has benefits and drawbacks. For example, SDN technology may be used to decrease or eliminate some of the security vulnerabilities and dangers that are frequently used against

conventional networks. However, the SDN technology's inventive design introduces additional dangers and vulnerabilities. [39].

All network security threats may be classified according to their primary purpose. For instance, listening in on a control interface can be categorized as an attack whose main objective is to interfere with private and sensitive data exchanged between network appliances; it represents an illegal disclosure of information[40]. Two features of SDN may be considered both appealing honeypots for malicious users and causes of annoyance for less experienced network operators. First, programs can be used to control the network. The controller's concentration of "network intelligence" (s) is the second. Anyone with access to the controller that hold the control software may possibly control the network [41]. All layers of the SDN network can be exposed to specific attacks that can expose the same layer or the entire network to penetration ;as it shows in figure 2.8.

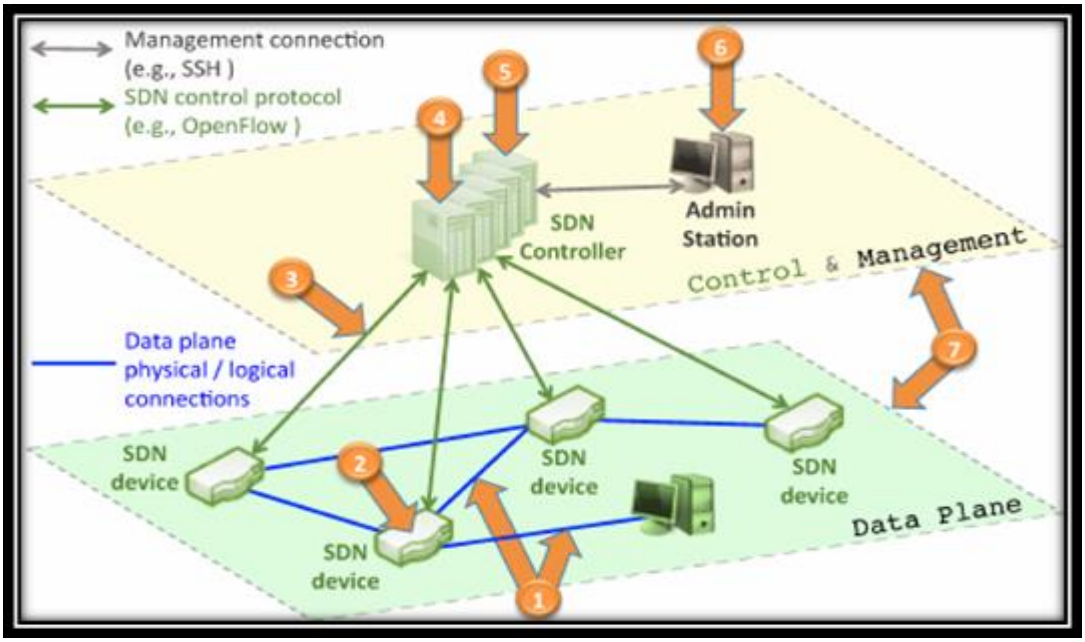


Figure 2.8: The weaknesses in the security of the SDN network [40]

The first type, of network weakness is faked traffic flowing to target switches and controllers. The attacker can exploit this by employing malfunctioning devices or network parts to execute an assault against switches and controllers [42].

The second type, is the attack on vulnerabilities in switches, which can lead to the destruction of the entire network. The attacker can use one switch to drop a packet from the network, change a specific path, or add fake requests so that more work occurs on the controller [43].

The third type, is a control plane attack for the purpose of launching a Denial of Service attack or data theft. In addition, TLS is not considered sufficient to ensure security between the controller and the switch, so once the attacker reaches the controller, the attacker will have the ability to launch a DDoS attack as well as leak data during the normal production flow [44].

The forth type, attacks and vulnerabilities in controllers, which are undoubtedly the most serious risks to SDNs. A defective or malicious controller could compromise an entire network. The deployment of a standard intrusion detection system might not be sufficient since it may be challenging to identify the precise set of circumstances that lead to a specific behavior and, more significantly, to identify it as malicious. Similarly, a malicious application may be able to do whatever it wants on the network because controllers simply offer abstractions that translate into giving the underlying infrastructure configuration commands [45].

The fifth Type, which appears due to the lack of methods and mechanisms that guarantee the relationship between the controller and applications, the controller and applications are unable to build trustworthy connections [46].

The sixth type, attacks and flaws in administrative stations, which SDNs utilize to reach the network controller. The network can now be reprogrammed from a single location [47].

And finally, in **the seventh type**, a lack of reputable forensics and remediation resources would enable understanding the root cause of an identified issue and moving forward with a quick and secure mode of recovery. It required trustworthy information from all network components and domains to investigate and establish the facts surrounding an occurrence. Furthermore, the value of this data depends on our ability to ensure its reliability (integrity, authenticity, etc.). Like prevention, restoration calls for secure and trustworthy system snapshots to provide network components with an efficient and accurate return [48].

2.2.3.1 SDN security countermeasures

The SDN security domain can often be divided into four major directions. First, research on importing security services that already exist. For instance, in [49, 50, 51] aimed to create and develop methodical ways for setting up dependable firewalls in SDNs.

Second, suggestions for enhancing current services using SDN's capabilities by incorporating its capabilities into standard security functions,

for instance, the authors in [52] explore whether SDN might improve network security. In a similar vein, approaches like [53, 54] that investigate how SDN might be utilized to safeguard networks against viruses also fall under this heading.

Recently, the emphasis has shifted to developing innovative security solutions that were previously impossible to provide. Examples include using network capabilities to safeguard clouds, smart grids, and Internet of Things (IoT) devices[55], [56].

In the fourth aspect research focused at protecting the SDN system which is crucial and directly impacts the adoption of SDN, makes up SDNs in the various tiers have been secured using a variety of ways. The goal of proposals like [57, 58, 59] is to secure controller planes.

Securing the northbound portion of an SDN is the focus of another area in the some research. For instance, it introduces a permissions mechanism to guarantee that only trustworthy apps can access controller operations. A crucial necessity is to secure an SDN's southbound. Securing SDN is divided into three categories: A) research aimed at safeguarding SDN's five primary components; B) research aimed at safeguarding its essential characteristics; and C) research aimed at safeguarding implementations Centralization and programmability that are the basic components of SDN and many of proposal have been suggested to protect them [60].

The implementation of SDN comprises protecting various controller platforms, the design and implementation of the OpenFlow

protocol, OpenFlow-enabled devices, and devices like Open vSwitch. Decomposing the literature in accordance with the SDN components that suggestions seek to achieve is an alternate method[61]. The review of the literature, however, shows that protecting the data plane of an SDN is one of the areas that has received the least attention. Therefore, this thesis aims to secure and protect the network by securing the data plane [62].

2.3 Cryptography process in SDN

The main principle behind the encryption process is that two parties wish to share a secret, without anyone else being able to understand it. [63]. This is accomplished through a process known as a cryptographic algorithm, in which Figure 2.9 depicts how to enter the plain text and the appropriate key for the encryption algorithm to calculate the cipher text and then retrieve the plain text in the reverse process by entering the cipher text with the key to option the plain text [64].

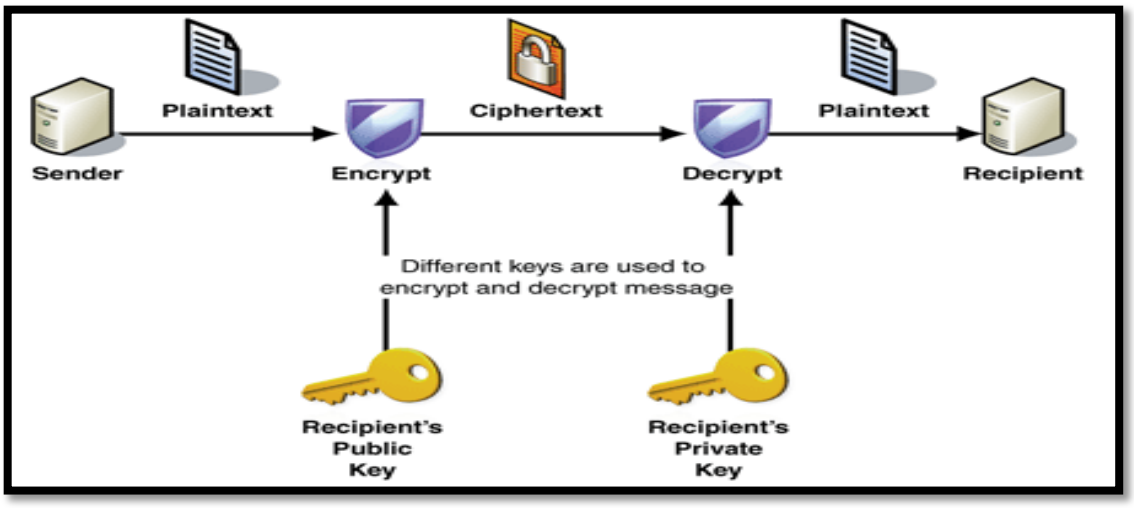


Figure 2.9: Encryption and decryption process[64].

2.3.1 Cryptographic Keys

In the field of cryptography, a key is a string of characters that is a component of an encryption algorithm that is used to make sensitive data appear incomprehensible. The cryptography locks data, meaning it is encrypted by using a specific key, and only the person who has the proper key may unlock (decrypt) it. There are two types of keys: the first is that the two parties use one key called the symmetric key, and the second is that each party has two different keys, one for encryption and the other for decryption, called a symmetric key [65].

1. **Symmetric Key Cryptography**

The main idea of a symmetric key is that the communication's two participants utilize the same key through the processes of encryption and decryption. Symmetric key cryptography uses straightforward and swift algorithms; this is necessary because the key must be delivered safely to prevent unauthorized access from a third party[66].

2. **Asymmetric Key Cryptography**

Asymmetric key cryptography, sometimes known as public key cryptography, requires that each party to a key pair have a public key and a private key. With asymmetric key techniques, it is possible to encrypt plaintext with the public key and decrypt it with the private key. Because any peer can share its public key, anyone can encrypt a communication that is meant for another peer. Asymmetric key techniques facilitate key distribution this is because there is no need for complex mechanisms, session keys, etc., as is the case with symmetric keys [67].

2.4 Key Management

The main goal of key management is to process the keys used in the cryptography process. The key management process relies mainly on the steps of the key life cycle, and the most important steps are key generation, distribution, and use. Designing a strong and secure key management schema is necessary because the management system relies on design rather than encryption algorithms [68].

In the next section, the basic stages of the key life cycle are explained, and then some of the basic management applications that were inspired by the management design used are discussed.

2.4.1 Phases of Key Management

These are considered the main steps in the implementation process, along with other secondary stages [69].

1. **Key Generation:** The key generation process is the first and most important stage in the life cycle of the keys. The process of generating keys using specific algorithms using random numbers to make the key more secure, the algorithms must correct the random selection. The speed of the key generation process depends on the type of key used, as symmetric keys are easier to calculate than asymmetric keys[69].

In the proposed model, the generation of the key showed in Algorithm 2.1 is delegated to the controller using the RSA algorithm..

Algorithm 2.1: RSA algorithm

Input: Two large prime numbers p and q

Output: A public key (n, e), private key (n, d)

Process begin :

1. Compute the product of the two prime numbers: n = p * q.
2. Compute the totient of n: φ(n) = (p-1) * (q-1).
3. Choose a public exponent e, such that 1 < e < φ(n) and e is coprime to φ(n).
4. Compute the private exponent d, which is the multiplicative inverse of(e modulo φ(n)) d ≡ e^(-1) mod φ(n).
5. The public key is the pair (n, e), and the private key is the pair (n, d).
6. To encrypt a message m, the sender calculates the ciphertext c as: c ≡ m^e mod n.
7. To decrypt the ciphertext, the receiver calculates the original message m as m ≡ c^d mod n.

End of algorithm

2. **Key storage:** Is a management process involving storing the required keys for later use. The storage location must be secure enough to avoid unwanted access[69].
3. **Key Distribution**: This is the process of transitioning keys to the intended entities in the network. The process of key distribution depends on the type of keys used. The distribution of asymmetric keys is easier because the process of symmetric key distribution requires a truly secured channel. Ensure secure access to nods so they are considered more difficult[69].

4. **Key usage:** It is a process in which the key becomes activated and available for use, and it follows the process of generation and distribution in the system[69].
5. **Key exchange**: a procedure wherein two reliable entities in a network decide on one or two keys to communicate between them[69].
6. **Destroying keys:** Is the process of removal of all copies of the keys, and this is done by deleting the entity that contains the key[69].

2.4.2 Architectures of Key Management

The primary management strategy is created based on the issue the system is intended to address. Due to the fact that the SDN environment always contains a central entity, the next section shows many examples of architectures that contain a central party [70].

- **Key Distribution Center (KDC)**

It is a method for distributing and generating keys between pairs of users to support random connections. Each user sends a unique key to the KDC, and as a result, the KDC issues shared keys for communication between entities. The master key is the key that the user sends to the KDC; the session key is the key that the KDC issues and is in charge of encrypting user communication[70].

- **Key Translation Center (KTC)**

This type is similar to KDC but does not create any keys. Before encrypting the subscriber's key with another key, the primary translation center decrypts it. The key shared with the KTC is used when any part of the network desires to deliver a message to another part of the network. The

KTC first decrypts the message and then encrypts it using the destination's key before transmission[70].

- **Public Key Infrastructure (PKI)**

A Public Key Infrastructure (PKI) provides asymmetric cryptography[71]. The administration of certificates is the PKI's duty in a system. Certificates perform asymmetric encryption and authentication by combining the public key and the owner's identity in a file authorized by the Certification Authority (CA). An entity must first confirm the validity of a certificate with a CA in order to communicate with the certificate's owner by using the public key, the organization then encrypts and sends the owner's private data [72].

In this architecture, the PKI provides the infrastructure for providing certificates to organizations that require them. Below is a list of the main components of a PKI[72]:

- CAs produce certificates based on previously established policies.
- Certificate Repositories (CR) hold certificates.
- Certificate Status Servers give information about certificates, including a list of revoked certificates.
- Registration Authorities (RAs) confirm the legitimacy of a business requesting a certificate.

2.5 Technologies of the SDN security

The following section provides an overview of the most important techniques used to implement the scheme.

2.5.1 RSA Certificates

RSA is a Public Key Encryption Algorithm that is used to keep sensitive data confidential. RSA was first presented by Ron Rivist, Adi Shamir, and Leonard Adelman in 1977. RSA uses two keys: one is a public key, and the other is a private key. Private keys must be kept secret, while public keys can be shared with anyone. The message is encrypted and decrypted with both keys using the RSA encryption formula [73].

One of the contributing factors to making the RSA algorithm so popular is that it provides a way to ensure privacy, integrity, and reliability. It is considered one of the safest algorithms due to the difficulty of decomposing the final number into two prime numbers. It is easy to multiply these two integers, but it will take a long time, even with supercomputers, to perform this analysis to extract the prime numbers from the sum [74].

Typically, a key is merely the information required for encryption. Additional information seen in a certificate includes the domain that the key is associated with, the organization, its expiration date, etc. To verify their integrity, certificates are typically signed with a separate key. Certificates are exclusively utilized in asymmetric cryptography and often only contain public keys [75].

The certificates are used in order to establish PKI. Lower-level certificates are signed using the private keys contained in key pairs contained in CA certificates. The certificate can be validated in this manner by validating a chain of certificates, where every certificate serves to verify the certificate immediately below it [76].

2.5.2 Open vSwitch

It is a virtual and open-source multi-layer switch that was used in this. It is designed to support massive networks through programmatic extension. It contains many tutorials, which makes it easy to use[77].

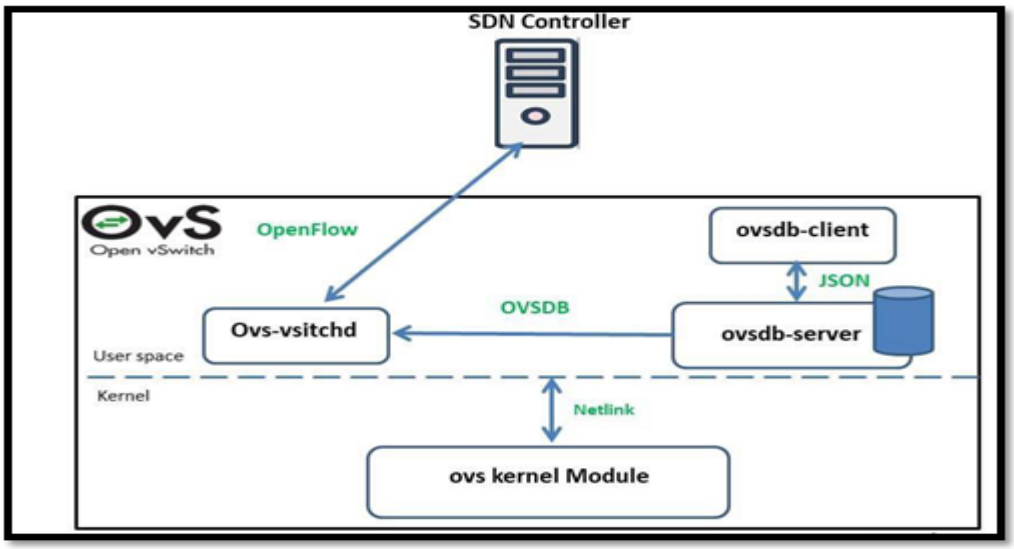


Figure 2.10: Open vSwitch Components [77]

Open vSwitch as shown in the Figure 2.10 consists of two main parts:

- The user space
- The kernel space.

The backbone of the open vSwitch implementation is OVS-vswitched. It uses OpenFlow to interface with the controller and adds several flows to the flow table in order to redirect data packets to the appropriate interfaces. Additionally, it uses the Open vSwitch Database (OVSDB) management interface to be able to communicate with the Open vSwitch Database Server (OVSDB-server). Through the netlink interface, the OVS

kernel model in the kernel space interacts with OVS-vswitchd in the user space [78]. The open virtual switch includes two important data forwarding modules: the first is the OVS-vswitched and the other is the OVS kernel module, as shown in Figure 2.11 [79].

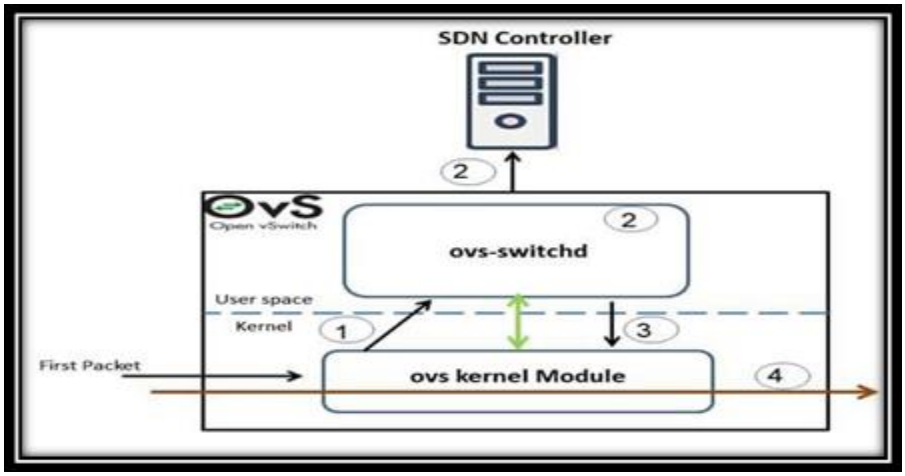


Figure 2.11: Forwarding data in OVS V-Switched[79]

The basic steps between these two main units in OVS-Switched are as follows:

1- The first packet to arrive from the stream will be sent to the controller. Either a controller is consulted, which in turn computes the route for the flow and sends actions (which may be one or more commands) to the ovs-switched component, or the ovs-switched component is manually programmed to calculate the route[79].
2- The instructions may include passing the packet to a specified port, encapsulating the packet and sending it to the controller, or finally deleting the packet[79].
3- Send commands to the OVS switch[79].
4- The rest of the packets belonging to the same stream are processed by the OVS Kernel component[79].

The OVS Switch works in two modes:

1: Standard Mode: It is the default mode; in this mode, OVS works as a traditional L2-switch (using the ARP protocol) in the process of learning and passing data between computers and therefore does not depend on the controller[80].

2: Secure mode: In this mode, OVS depends on the controller in the process of assigning streams to pass data[80].

2.5.3 RYU Controller

The RYU Controller is entirely written in Python, which is backed and employed by NTT cloud data centers and is open source, provided under the Apache 2.0 license. The Open RYU community has donated and is supporting the main source code, which is available on GitHub. In addition to OpenFlow, it supports the network management protocols NETCONF and OF-config. RYU controller designed to increase network speed and facilitate data traffic. It is the brain of the network and is responsible for communicating all the data via API interfaces [80].

RYU controller like other SDN Controllers, generates OpenFlow packets and oversees events pertaining to the incoming and outgoing packets. A long collection of libraries that handle packet processing activities is available. RYU is collaborating closely with protocols like OVSDB, OF-Config, NETCONF, and XFlow (Netflow and Sflow) to provide southbound protocols. RYU Packet Libraries also support VLAN, GRE, and other similar technologies [81].

Many other programs, including a simple switch, router, firewall, VLAN, etc., are distributed with RYU. Applications written in RYU are

single-threaded objects that implement a variety of capabilities. Each other receives asynchronous events from RYU applications. Figure 2.12 depicts the system design of a RYU application. There is a queue first-in first-out (FIFO) in every RYU application for events in order to maintain the order of events. The thread's main loop runs the necessary event handler while pulling events from this queue. Due to the blocking nature of the event handling thread, which executes the event handler within its scope, no other RYU app operations will be processed until control is regained after an executable has been given control [82].

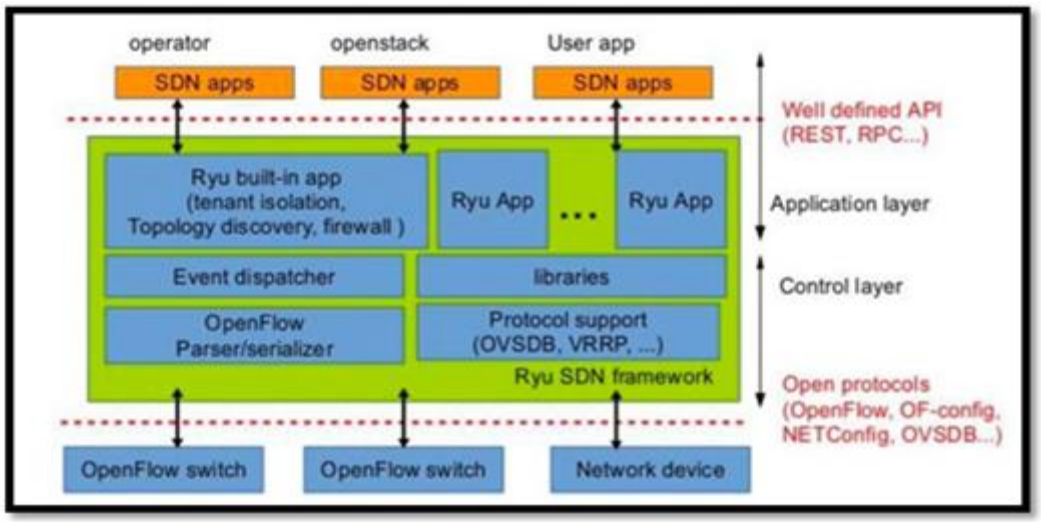


Figure 2.12: RYU Controller [82].

2.6 Evaluation Metrics

The proposed system is evaluated with the different evaluation metrics as follows :

2.6.1 Throughput

Throughput is described as the ratio of the number of packets sent and received to the total required time. With the corresponding equation being [83] :

$$\text{Throughput} = \frac{\text{Total Number of sent and received packets}}{\text{Time}} \quad (2.1)$$

2.6.2 Randomness Key

It measures the degree of (randomness) or disorder in a system. The randomness key of the uncertainty of the random variable (X) with probabilities (pi, …, pn), log means natural logarithm, as defined in equation (2.2) [83].

$$Randomness\ Key\ Test(x) = -\sum_{i=1}^{n} P(i) \quad log_2(P\ (i)) \quad \ldots. (2.2).$$

2.7 Implementation Tools

The proposed system is implemented in Mininet and Wireshark .

2.7.1 Mininet

Mininet an SDN simulation tool for creating a number of virtual hosts, switches, controllers, and links. The main features of Mininet are to provide a simple and fast way to implement and evaluate console application and SDN protocols. Mininet-Wi-Fi is a fork of Mininet that allows the use of both Wi-Fi stations and access points. Mininet-Wi-Fi adds Wi-Fi functions and currently Mininet-Wi-Fi emulator uses standard Linux TC tools to imitate the wireless channel by adjusting link characteristics such as packet loss, delay, and channel bandwidth. Mininet allows rapid prototyping of a large virtualized network infrastructure using only one common computer in a virtual environment. Using simple software such as OpenFlow, we can develop virtual prototypes of scalable networks that operate like physical networks based on OpenFlow. Incorporating these core building blocks provides you with a foundation for creating, interacting with, and customizing software-defined networks quickly. Some features of Mininet[84]:

- Simplifies and streamlines the testing of OpenFlow APIs.
- Enables independent researchers to do independent network configuration activities on the same network architecture.
- Offers the ability to test sophisticated and massive topologies without requiring a physical network.
- Supports testing via the network with tools to troubleshoot and to perform the tests.
- This provides a broad range of topologies, from the basic set. Networking is made simple by offering a simple Python API for constructing and testing networks. The official website is : http://mininet.org/

2.7.2 Wireshark

It is an absolute and unlock source packet analyzer used commonly in networks for packets management especially in SDN networks. The packets in this tool can be captured directly from the network and displayed in a human-readable format. It is used for network troubleshooting, analysis, software development, communication protocols, education, filtering packets based on special criteria, generating statistics, etcIn the first creation of this program, it was called Ethereal, and in May of the year 2006 it was called the current name due to trademark problems[85]. The official website is : https://www.wireshark.org/.

Chapter Three

Research Methodology and Proposed System

3.1 Introduction

This chapter provides details of the suggested model of the cryptographic key management project in the software defined networking (SDN) environment that has been developed to design and implement a secure and efficient system for managing cryptographic keys and also describes the design of the proposed system in detail. Each part has a different function integrated to protect the data. A brief description of the steps of the proposed system will be included in this chapter.

3.2 Proposed System

The proposed system is represented in procedures to achieve a secure distribution of keys and to manage those keys automatically in order to use them to encrypt data sent over the network. In the next section, each step of implementing the proposal is discussed and explained using tools and algorithms. Figure 3.1 showed the proposed network architecture. Besides, the general approach and sequence of steps followed in this thesis are depicted in Figure 3.2.

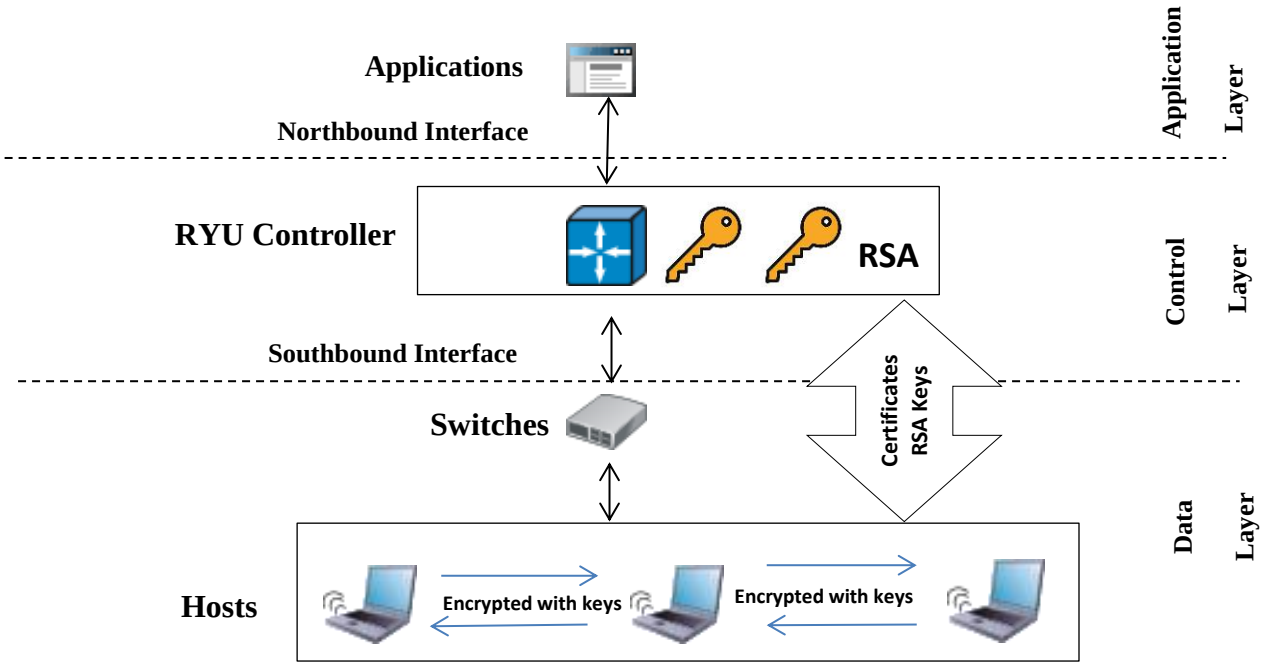


Figure 3.1: The proposed network architecture

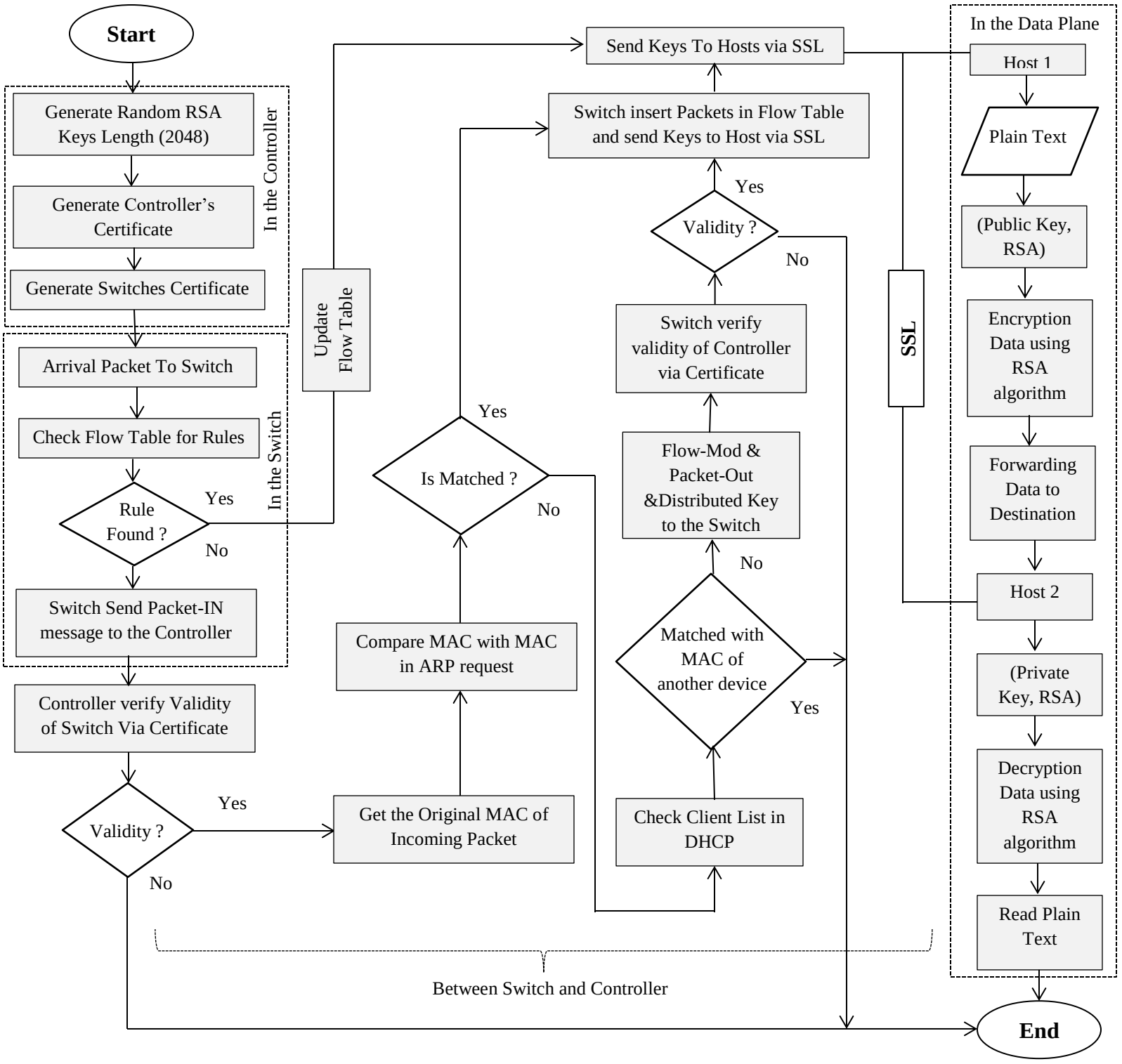


Figure 3.2: Block Diagram of Research Methodology and Proposed System.

3.2.1 SDN Setup Procedure

Establish a Public Key Infrastructure (PKI) using VMware workstations as a virtual simulation program and start the process of implementing the scheme by creating keys, certificates, and a network topology. An SSL channel is then formed between the controller and switch to send the keys for later use in the process of encrypting the data

Steps of setting up SDN infrastructure:

1. **Establish a Public Key Infrastructure (PKI)** that contains two subdirectories:

A. Controller CA

- Cacert file: This file contains the root certificate to verify the validity of the control. A copy of it is sent to the switch to authenticate valid controllers.
- Private CA key: This file contains a private key that is used by the CA to sign the authenticating controller.

B. Switch CA

- Cacert file: This file contains the root certificate to verify the validity of the switch. A copy of it is sent to the controller to authenticate a valid switch.
- Private CA key: This file contains a private key that is used by the CA to sign the authenticating switch.

2. **Create a controller's private key and certificate:**

- Use the command "ovs-pki req+sign ctl controller."
- Copy the "ctl-privkey.pem" and "ctl-cert.pem" files to the controller to use itself at another communication.

3. **Create a switch's private key and certificate:**

- Use the command "ovs-pki req+sign sc switch" to generate the "sc-privkey.pem" and "sc-cert.pem" files.
- Copy both files and the "cacert.pem" file to the Open vSwitch.

4. **Configure SSL support:**

- need unique "sc-privkey.pem" to add the private key of RSA algorithm and "sc-cert.pem" to ensure the validity of private key and also use the "Cacert file" to the validity controller.

5. **Create a new bridge.**

- Create a new bridge to connect the switches together and establish a virtual network.
- Configure the bridge with parameters such as the name, type of controller to be used, and physical ports to be connected to the bridge**.**

6. **Create a network topology and select number of device.**

A- Controller B- Switch C- Nods

7. **Establish an SSL connection:**

- Use the SSL certificates to establish secure communication channels between the controller and switches.

8. **Start Communications**

3.2.2 Establish a Public Key Infrastructure (PKI)

It is a set of devices, services, policies, and some procedures necessary to create digital certificates and their keys and to control the process of distributing

and using those certificates. It allows the use of important technologies such as encryption and digital signatures between a large group of users. The PKI helps identify users within the network and protect data by adding some details and making access to data and systems under control, as shown in Figure 3.3.

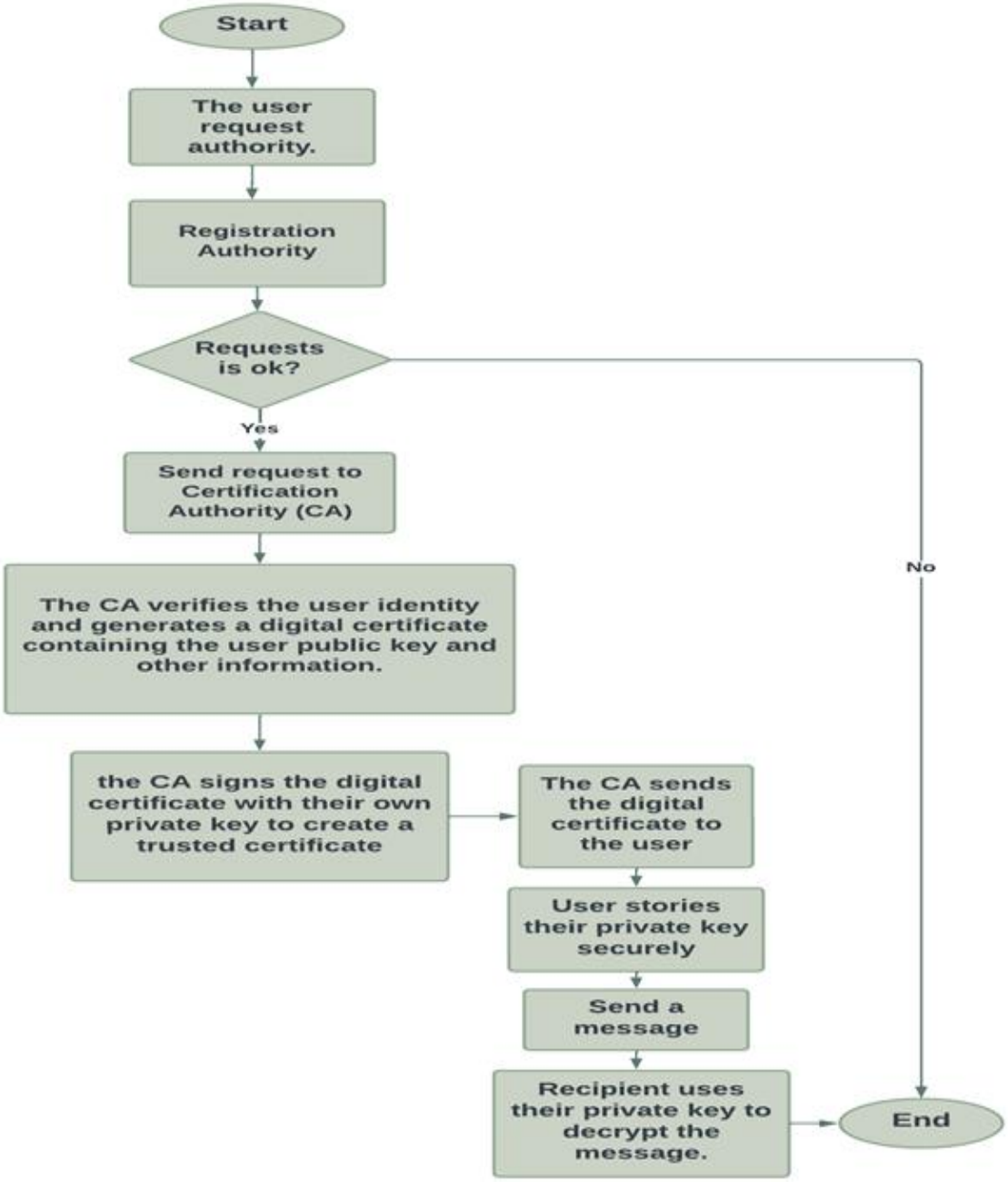


Figure 3.3 : FlowChart of PKI

3.2.3 Generating Keys

There must be some sort of algorithm that compiles random keys to produce the keys in this key management system. When it comes to key generation, there are two options: either the controller creates keys before sharing them, or the nodes create keys on their own and transmit the controller their public key after.

If the node generates the pair of keys**;** the node is responsible for the way in which it generates the keys and for ensuring the strength of those keys. First, it must make sure that the private key is stored in a safe place and cannot be accessed easily. It is also responsible for obtaining authentication of the public key by the certification authority by presenting the same node to the certification authority. To have public key authenticated by a CA, a user can provide himself and public key to the CA. The user is then authenticated by the CA.

If the node is authenticated, the CA will perform tests on the key that was provided to ensure its strength and also to make sure that it was actually presented by the node that was registered. The CA offers the node a certificate that links their identity to their public key after confirming both the validity of the key and their identity.

In the paragraph explained above, which states that nodes generate a pair of keys, it is assumed that the node has certain software or hardware capable of generating those keys. This method may be impractical in terms of cost and also in terms of time if the number of nodes in the network increases. For this reason, it is necessary to have a central system or part that is responsible for creating the keys, and here comes the role of the controller. Where the controller and the certification

authority can be in separate places. To obtain the key pair, the RSA algorithm inside the controller is used to generate a key pair.

When a new request comes to the controller, the user is given both the public and private keys by the system when it generates the key pair. After providing the user with the private key, the controller promptly destroy all copies of the key. The controller needs to do more than just provide the CA the public key in order to receive the certificate.

3.2.4 Generating Certificates

In a key management system, a key certificate is a digital document that verifies the identity of a specific public key's owner and is built on a specific structure that contains some information. It provides proof that the public key belongs to a specific individual or organization and is used to prove the public key's authority as being trustworthy.

To generate the key certificate, Digital signatures are created by the public key's owner using their private key. The digital signature is then attached to the public key and other relevant information, such as the owner's name and contact details. After that, Certificate Authority (CA) receives this data package for verification. The CA authenticates the public key owner's identity before using its own private key to sign the key certificate. The signed key certificate is then returned to the owner, who can use it to prove their identity when communicating with other parties. RSA is commonly used in the generation of key certificates because of its strong security properties, as shown in Figure 3.4.

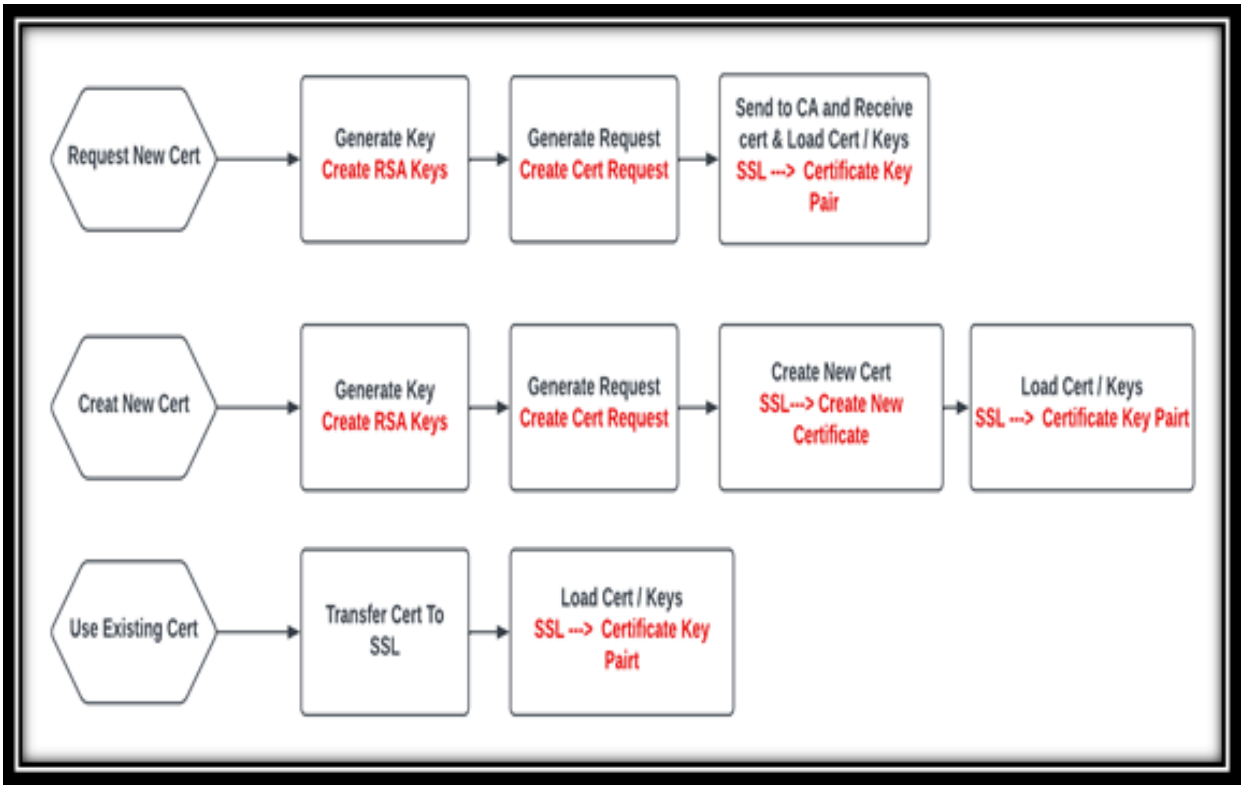


Figure 3.4: Flow Chart Certificates

RSA is a public-key cryptosystem that allows for the secure exchange of information by employing a public key to encrypt data and a private key to decrypt it. This makes RSA ideal for generating digital signatures, which are a key component of key certificates. When a certificate arrives for the devices, the system proceeds to the device that has already made a request. The device retrieves the key from the certificate to encrypt the message before send it.

The following details are included in the format of SSL certificate:

Certificate:

Data:

Version: 1 (0x0)

Serial Number: 1 (0x1)

Issuer: C-US, ST=CA, 0-0pen vswitch, OU-controllerca, CN-OVS controller

Validity

Not Before: Feb 25 09:48:34 2023 GMT

Not After : Feb 22 09:48:34 2033 GMT

Subject: C=US, ST=CA, 0-0pen Switch, OU-controllers, CN-OVS controllers. Example bellow showed the RSA cryptography algorithms inside data structure of certificates.

Example for Public Key Algorithm based on RSA encryption

Input : RSA Public-Key: (2048 bit)

Output :Cipher

Modulus:

00:cb:94:b3:2:8c:78:aa:78:e6:58:a0:7c:ce:44:75:4:3c:a6:7b:20::64:1:9:51:0d:87:4ab:4a:f1:dc:5c:5e:06:cf:e8:0f:f2:6e:08:2:3:de:65:77:4:fb:ff:b5:47:5f:42:ea:78:90:c:a0:05:4:da:6d:28:67:17:97:7d:2:8d:9:ce:7e:2:8d:1b:21:88:22:16:77:ab:ff:78:00:cb:94:b3:2:8c:78:aa:78:e6:58:a0:7c:ce:44:75:4:3c:a6:7b:20:83:64:1:9:5c:5e:06:cf:e8:0f:f2:6e:08::97:7d:2:8d:9:ce:7e:2:8d:1b:21:88:22:16:77:ab:ff:78::fb:ff:b5:47:5f:42:ea:78:90:c:ab:cc:77:33:af

Algorithm: RSA Encryption

48:83:77:ae:6b:95:54:60:b6:6b:ad:5c:b2:07:43:0b:ab:22:f4:65:43:00:e4:3d:8c:d4:44:2b:12:1e:3a:2c:79:e7:84:bb:be:bc:74:27:13:73:57:66:40:5e:53:27:80:55:37:d7:d0:d0:60:6b:ae:73:75:8:d8:4e:e7:08:b6:38:38:e3:b8:1f:e8:cb:66:8c:24:26:c3:a3:40:cC:8d:f3:25:71:e4:c3:29:ac:2e:a0:f9:a1:52:56:2d:73:3c:e1:20:77:13b4:85:60:a3:88:24:f2:a7:eb:6a:f3:bc:6b:97:fa:98:97:39:0a:2c:6:17:86:6d:a5:a8:6c:be:ae:71:d7:d4:2c:38:4e:3e:ae:85:3c:44:be:12:3d:9e:e2:0c:3c:bf:82:0e:ce:58:84:c9:df:09:8b:4f:a5:67:7f:a6:73:1:c2:8a:ab:7e:cd:2e:3e:88:8c:f8:23:be:07:40:9e:ac:b8:cd:f6:f5:e5:e1:ad:08:a1:1c:ab:e5:d9:ba:ff:37:4c:d2:8d:73:a3:05:1c:3e:04:f5:ff:79:12:89:80:6c:10:73:04:20:a4:18:2:2d:e4:cb:6a:cf:e8:b3:d825:34:56:3c:61:be:03:C4:9a:88:d9:d5:2f:ce: 45:f9:6d

To clarify the above details:

- Version: certificate data format's version number.
- Serial number: Certificate's unique identifier as assigned by the CA.
- Issuer: the certificate's issuing authority's name.
- Validity: It contains two periods. The first one is valid-From: The certificate's validity period, and the other's is valid-To date is the expiration date.
- Subject: Domain name, country, and name of the owner.
- Public Key: The owner's public key.

3.2.5 Key Storages

The key storage process generally depends on the mechanism used to generate the keys. There are three cases for the key storage process, namely: Either the controller maintains no keys, only public keys, or both public and private keys are stored by the controller. Due to the SDN environment's provision of the controller as an authority for both nodes, the controller is depended upon for storage.

- **No keys stored on the controller:**

The Key Translation Center (KTC) architecture, which is explained in section (2.4.2), would be the one that is employed in this way. The controller acts as KTC, and each node would be connected to a secure OpenFlow channel for the transit of keys. This approach is not requiring the distribution of private keys to the controller because the keys must generate by the nods.

- **Only Public keys are stored in the controller:**

Public keys are kept by the controller. By maintaining an updated list of each of the public keys associated with each node, the controller, in this way, works as a public key authority, as described in section (2.4.2). After that, the process of creating keys begins, either by the controller or by the node itself. The controller would distribute keys faster than if no keys were stored. This approach is considered more secure because only the nodes that truly require the keys have the ability to access them.

- **The controller store both the public and private keys:**

The controller, in this way, will be able to produce the private and public keys and then distribute these keys to the associated node. This method is considered one of the best and fastest methods previously mentioned, but it contains a problem, which is the presence of the private key in more than one element, so this problem was solved by using certificates. The third way—the Controller keeping both public and private keys—was selected for the suggested model.

3.2.6 Generating Network Topology

There are many types of SDN networks, as shown in Figure 3.5. Each type differs from the other in the number of devices, switches and the controllers, also according to the need and requirements of work.

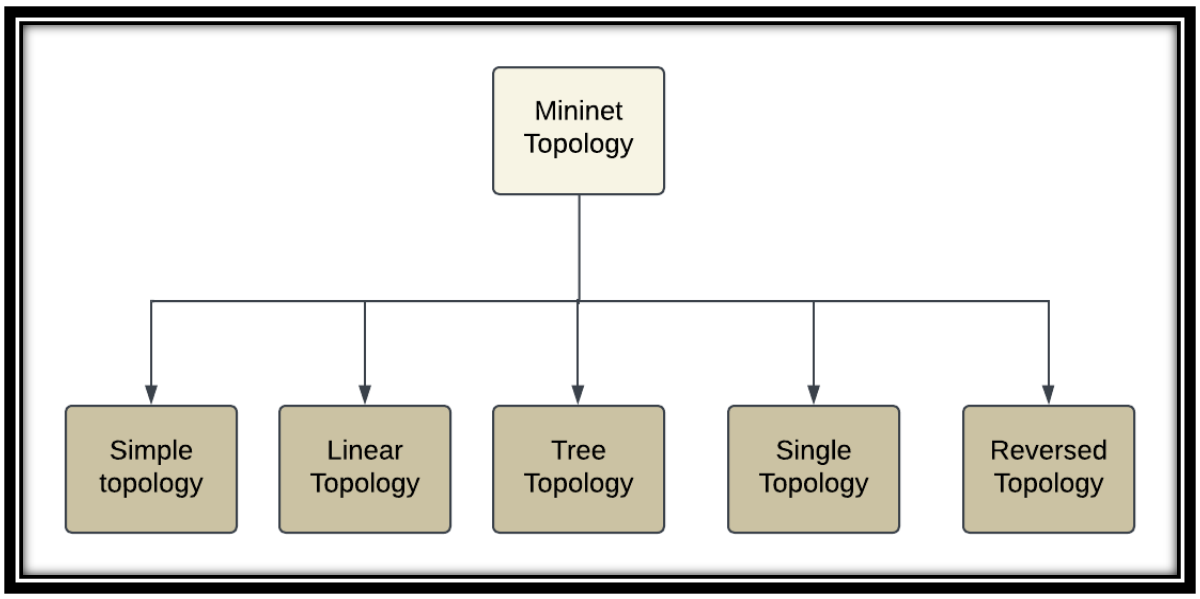


Figure 3.5: Main type of SDN topology

The network type chosen in the proposed work is a single type, as it contains one controller (RYU type), one switch (Open VSwitch type), and three hosts, as show in Figure 3.6.

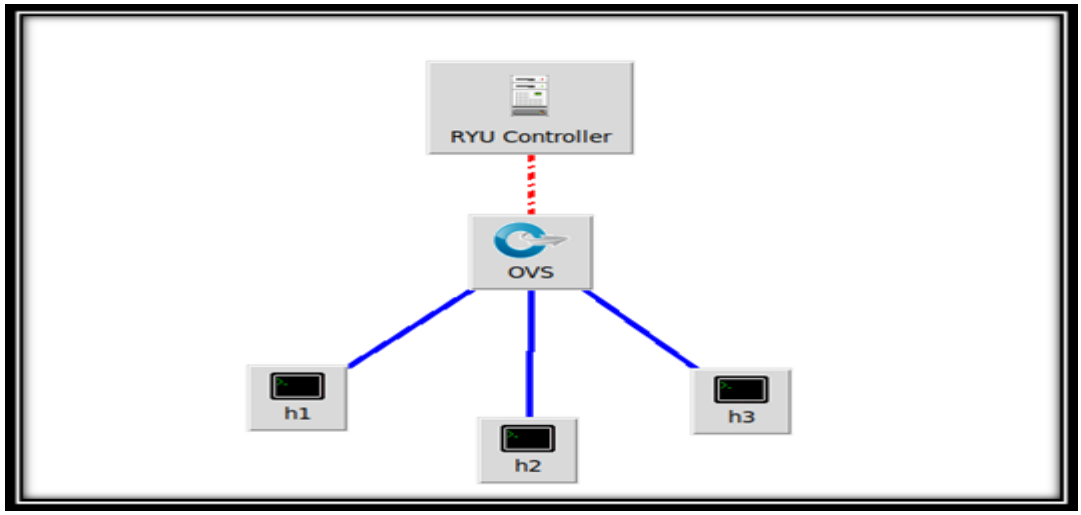


Figure 3.6: The proposed network topology.

3.2.7 Ensure The Authority of The Devices

The system checks whether the request is coming from a trusted device by going through two stages, one of which is related to the other, namely:

The first method of the detection process by using Physical MAC

When a node has the destination's IP address and wants to learn the Media Access Control (MAC) address of that location, the ARP is used inside the network. If the destination hasn't been registered before, a controller that gives it access to the necessary interface is needed. The sender sends an ARP request packet to discover the interface's MAC address if the sending device's ARP table does not already have it. The attacker observes this request and responds on the victim's machine (the sending machine) with a fake ARP response packet. The most recent response supersedes the oldest in the ARP protocol. As a result, the spoof IP/MAC mapping will be saved in the target machine's ARP cache. Where a method was used to detect and prevent this type of spoofing, as shown in the Figure 3.7.

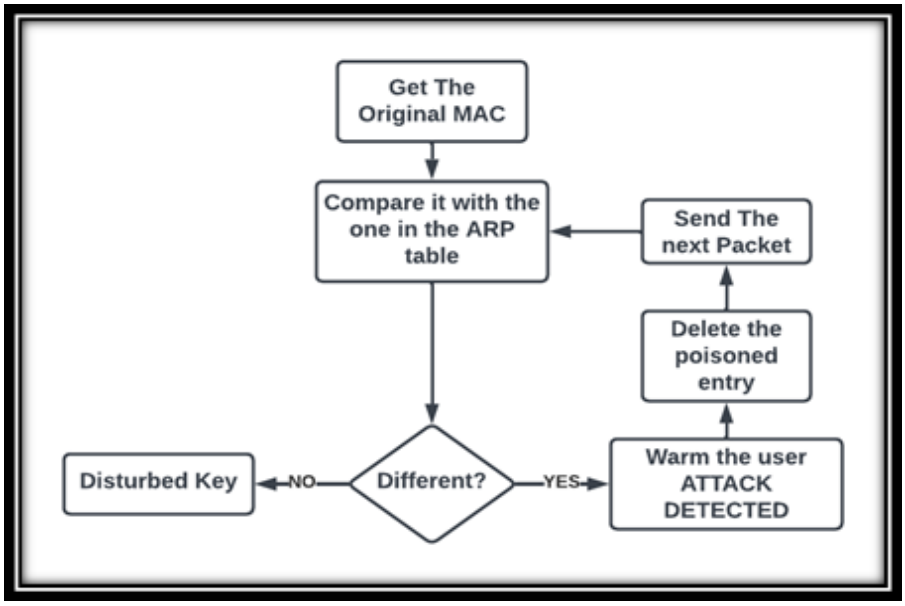


Figure 3.7: The used method to detect and prevent attack

The second method of the detection process is by using the DHCP server.

When the device accesses the network, it takes an IP address through the DHCP server located inside the controller. After going through the first stage, the packet now enters the second stage of verification, and this is done through the table list inside the DHCP server, which contains all the original MAC addresses of devices that were previously registered when the device was connected to the network for the first time. If the MAC address coming from the requesting device matches one of the previous addresses of another device, then the device is not trusted, and if the MAC address in the request matches its MAC address in the MAC list, the controller considers it a safe host. Figure 3.8 shows the second method for detecting the presence of spoofing.

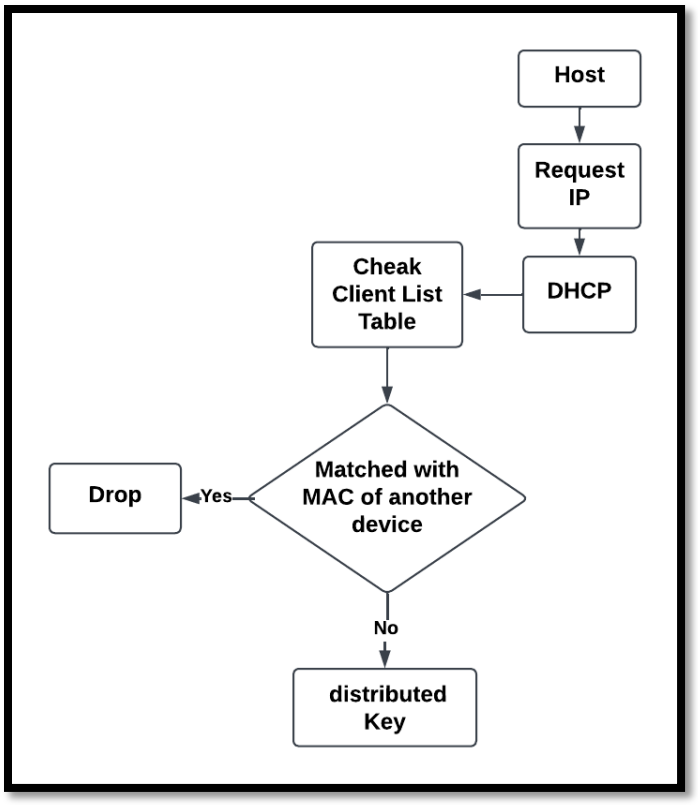


Figure 3.8 : The proposed second method to detect and prevent attack.

3.2.8 Key Distribution

In the context of SDN, key distribution is the process of distributing public and private keys to network devices to enable secure communication and data exchange. RSA is a cryptographic algorithm used, also called a public key cryptographic system. The process of generating and distributing keys to network devices in SDN implement by the controller. In the proposed scheme, public keys were used to encrypt messages and private keys to ensure that messages are not decrypted except by the intended recipient, as those keys are distributed using a secure channel that helps prevent unauthorized access to the keys. Figure 3.9 shows the mechanism of distribution of keys.

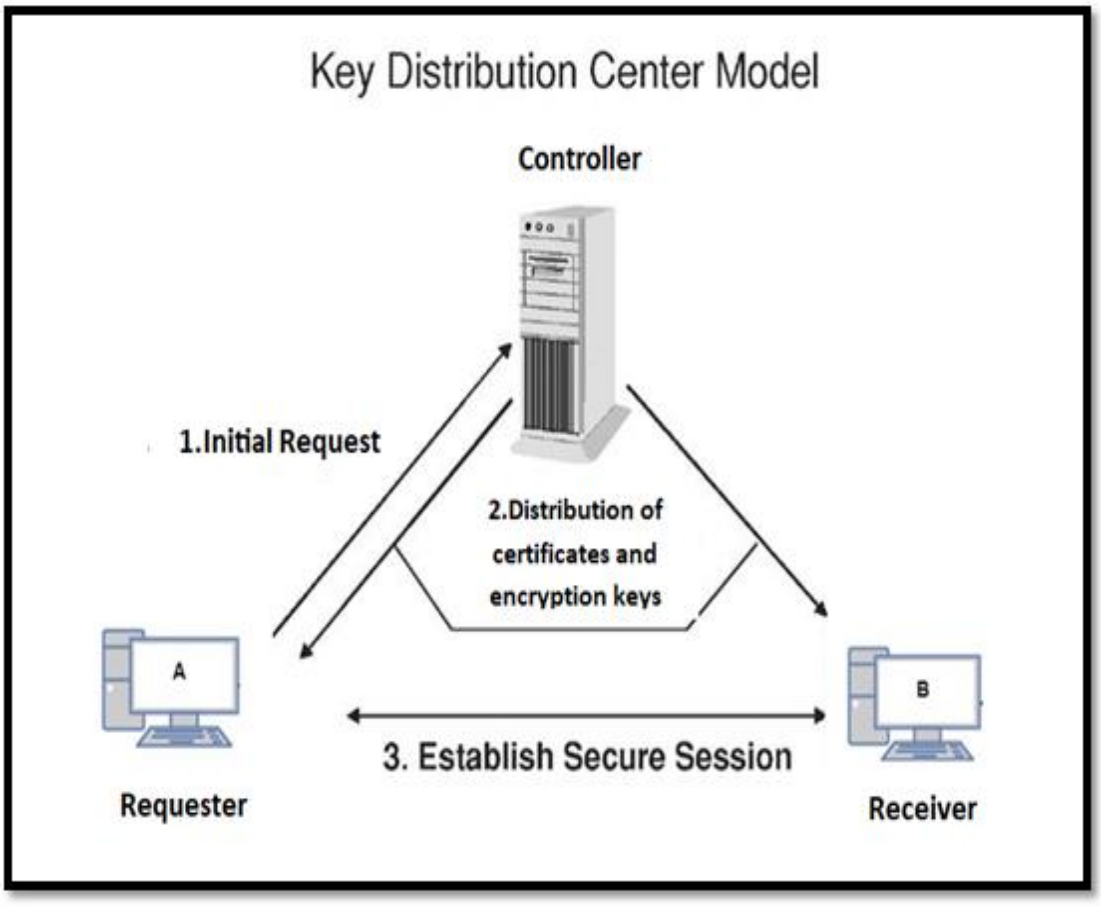


Figure 3.9: Mechanism key distribution

There are two modes for the distribution of the key:

1- Reactive key distribution: denotes the distribution of keys in response to management system requests. The keys are created and then distributed to the chosen nodes to begin encrypted communication.
2- Proactive key Distribution: In a proactive key distribution system, the controller fills the nodes' tables before encrypting communication.

Because the SDN architecture is highly dynamic, the first method was relied upon when designing the scheme.

Key distribution data structure refers to the way in which cryptographic keys are distributed and managed in a secure system. There are various ways to distribute cryptographic keys, and the choice of data structure depends on the specific needs and requirements of the system. One of the data structures commonly used for key distribution is the Public Key Infrastructure (PKI), which operates on the basis of the existence of digital certificates. In a PKI system, a trusted third party, known as a certificate authority (CA), creates digital certificates for users, which contain the user's public key and other information. When a particular user needs to communicate securely with another user, they can use the recipient's public key, which is included in their digital certificate, to encrypt the message. When the message reaches the recipient, then they can use their private key, which is kept secret, to decrypt the message.

3.2.9 Using Keys Within RSA Algorithm

After the keys are distributed and stored, they can be used for secure communication and data exchange between SDN components. The use of keys ensures the encrypt and decrypt data, authenticates network components and

secures data traffic. When the certificates and keys reach to the sender, the following procedure is happened :

- **Sender Side:**
- Encrypts the message using the keys before sending it to the destination.
- Verifying with authentication process.
- **Receiver Side:**
- Verifying authenticate user
- It receives the encrypted text message (cipher), it can use the same encryption algorithm and specialized key to decrypt the message and restore the original plain text message as show in Figure 3.10. This ensures that the data remains confidential and invisible to parties who do not have access to that data, providing an additional layer of security for an SDN

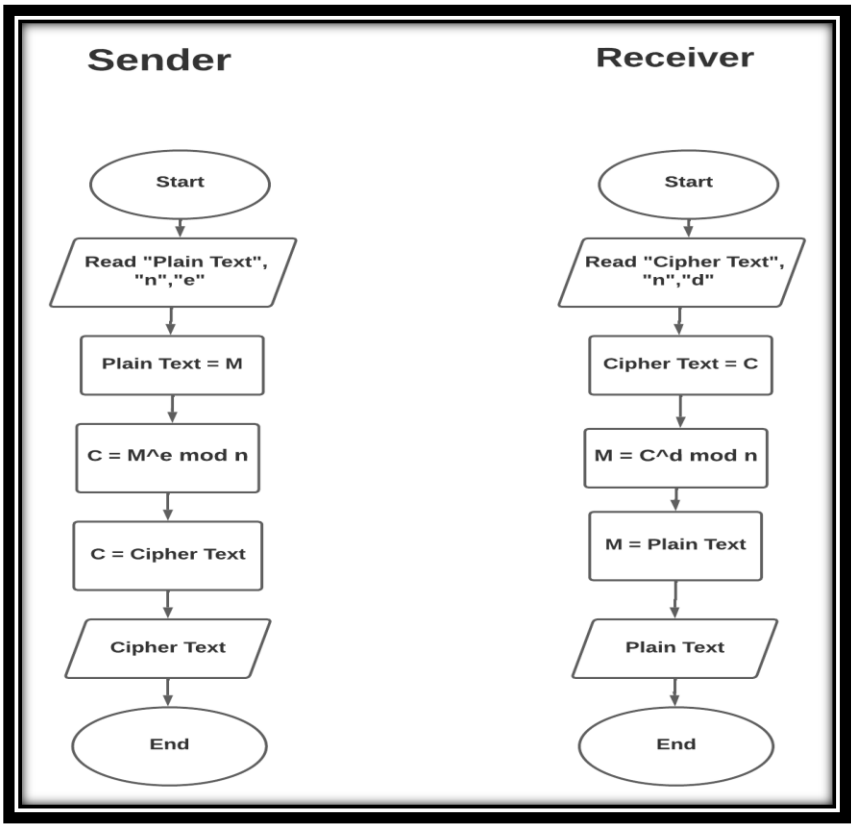


Figure 3.10: Encryption And Decryption Message.

3.2.10 Communication Mechanism

Figure 3.11 shows the communication process between the control plane and the data plane and the messages sent between them.

1- Host A sends the packet stream through an ARP request message to specify the logical address of host B.
2- The OVS switch receives a copy of the ARP request packets via port 1 and then sends a Packet-in message to the controller containing the ARP request.
3- The controller knows from the Packet-in header that host computer A is on this port 1, so firstly, check the authorization of the device to ensure the device is not spoofing.
4- The controller sends a Flow-Mod message to insert the entry into the flow table, which will help the switch to send packets destined for computer A to port 1 in the future without the need to consult the controller.
5- The controller sends a Packet-Out, which tells the switch to send an ARP Request message to all ports (so that it can find out the logical address of B).
6- The switch sends an ARP request to all ports to which it is connected directly (in this case, it has port 2 only).
7- Computer B receives a copy of the ARP request message and replies with an ARP response message.
8- The switch sends a Packet-in message to the controller containing ARP Reply.
9- The controller knows from the Packet-in header that host B is on port 2, so firstly, check the authorization of the device to ensure the device is not spoofing.

10- The controller sends a Flow-Mod message to enter the flow table, which helps the switch by sending Packets destined for computer B to port 2 in the future without the need to ask the controller.

11- The controller sends a Packet-Out, which tells the switch to send an ARP Reply message to computer A through port 1.

12- After the controller verifies the reliability of the devices in step (3,9), it distributes the encryption keys to the devices using certificates to encrypt the messages sent between the two parties.

13- Computer A receives an ARP Reply message and keeps B's physical address in its ARP Table (thus, now it can compose a packet to be sent by computer A).

14- Computer A uses a public key to encrypt data and sends a ping request to computer B.

15- The switch performs a lookup in the Flow Table and passes the packet according to the input (it passes the packet to port number 2).

16- Computer B uses a private key to decrypt data sent from host A and read the data.

17- Host A also use public data to send a ping Reply.

18- The switch performs a lookup in the Flow Table and passes the packet according to the input (passes the packet to port number 1).

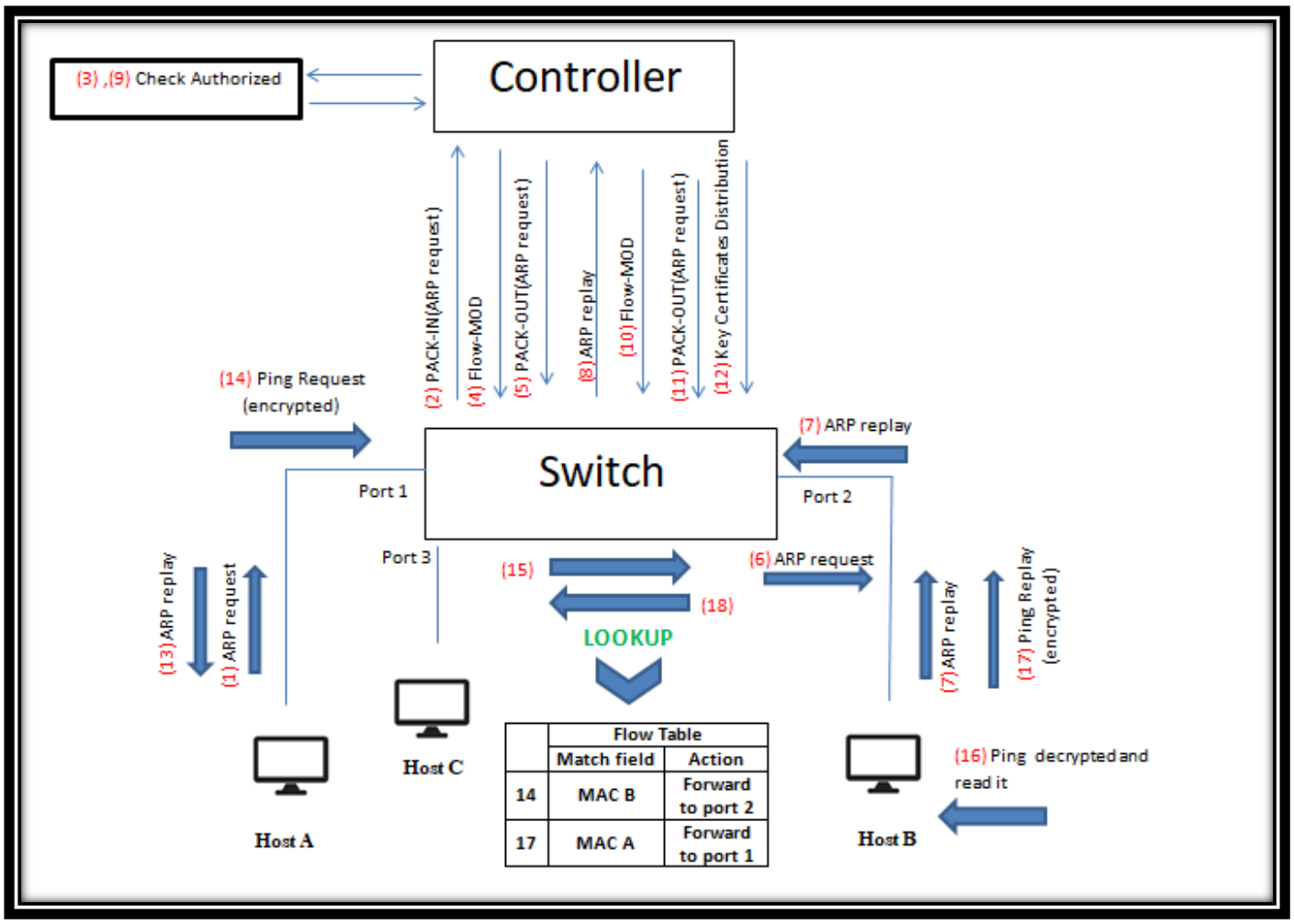


Figure 3.11: Communications through ARP request message

Chapter Four

The Implementation and Results

4.1 Introduction

In this chapter introduces the simulation and discussion of results of suggested Cryptography Key Management in Software-Defined Networking(SDN) , and it has simulated by using VMware built in a python programming language to implement the network performance in an adaptive way.

The presentation starts with establishing the RSA algorithm, generating the public and private keys, creating and configuring certificates, storing them inside the controller, and then creating a network consisting of three hosts and one switch connected to each other by a physical link cable and connecting the switch by a logical cable with the controller. Finally, the communication process begins, the keys are distributed to the network devices in a secure manner, and the detection application is activated to prevent the random distribution of keys to unauthorized devices.

4.2 Simulation Environment

Mininet was used as a simulation of the work environment, and a RYU was used to represent the controller, and then they were installed inside the Linux-Ubuntu with VMware tool-box. In this experiment, an Open Flow switch or what is called Open vSwitch was used by activating the Open Flow protocol. The network was programmed using Python as a programming language. Table 4.1 presents the simulation setup parameters.

Table 4.1: Simulation set up parameters

Parameter	Value
The simulator	Mininet
Controller	RYU
Switch	Open virtual switch
Cryptography Algorithm	RSA

There are several important aspects that need to be evaluated when implementing a proposed key management schema to be knowing how it is affecting, such as hardware resources needed by the network, the percentage of accuracy in the process of detecting the spoofing device, in addition to evaluation of network performance in terms of throughput.

4.3 Hardware resources result

One of the most important aspects that must be considered first is the amount of resources needed to carry out any work in order to ensure that all the necessary needs are provided to avoid any defect during implementation.

In the beginning, the environment in which the work was executed was measured and evaluated to determine the amount needed for the work in terms of memory and CPU, as well as in terms of a hard disk. The initial values of the parameters were taken before execution, and the values as follows:

Table 4.2: The initial value before implementation

Parameter	Values
Memory (m1)	4.73 GB
CPU (c1)	1.70 GHz
Hard Disk (h1)	19.1 GB

After that, the network was configured and set up, and the scheme was implemented. After that, another sample of parameter values was taken after completing the implementation process, and the results of the values were as follows:

Table 4.3:The value after implementation

Parameter	Values
Memory (m2)	3 GB
CPU (c2)	100MHz
Hard disk (h2)	900 MB

Through the results obtained before and after implementation, the total values of the resources that were used or needed in the implementation process can be extracted below and shown in Figure 4.1:

Table 4.4: The Final resource used during implementation

Parameter	Values
Memory(m=m2-m1)	(4.73 GB – 3GB) = 1.73 GB
CPU (c=c2-c1)	(1.70GHz – 100MHz)= 1.6 GHz
Hard disk (h=h2-h1)	(19.1 GB – 900 MB) = 18.2 GB

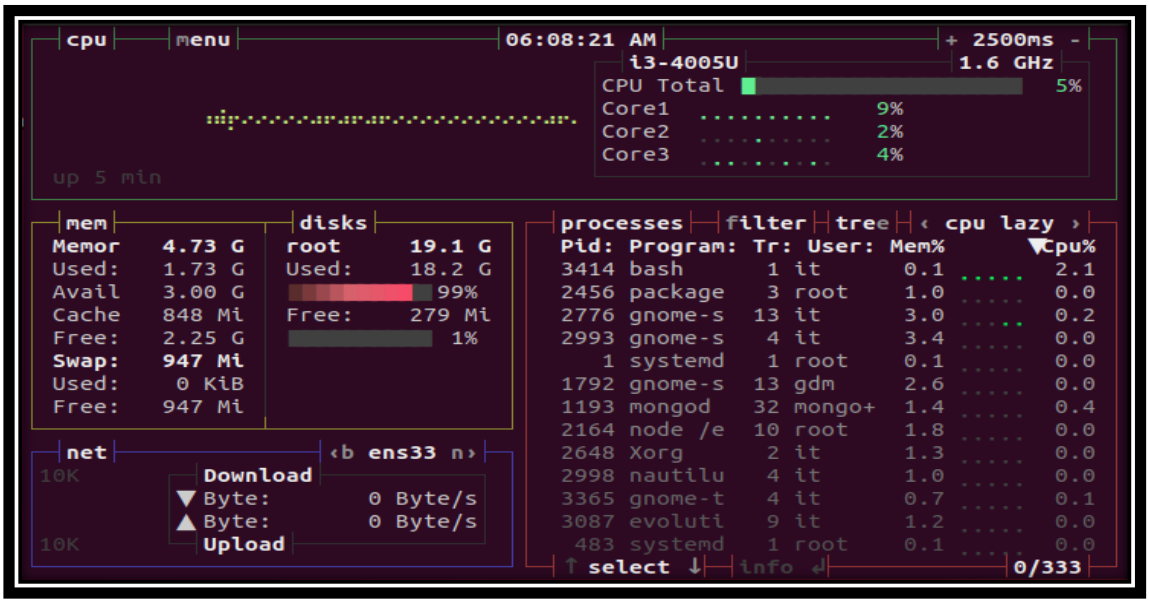


Figure 4.1: The final results of parameter

4.4 Generate Certificates and Keys Result

First, public key infrastructure must be created in order to be able to properly configure and manage the certificate to be able to create an initial PKI. This will, as shown in the Figure 4.2, final result of populate new PKI files. The configured PKI directory contains two main directories, controlerca and switchca.

Figure 4.2: Generating PKI

4.4.1 Implementation within the Controller

After the PKI had established, keys and certificates are now generated for the controller.

1. Creation Keys Result

The process of generating keys is implemented using the RSA algorithm with a length of 2048 bits to produce the public key and the private key of the controller for later use in encryption operations where the private key is first generated, as shown in the Figure 4.3.

Figure 4.3 : Private key of Controller

It must also be ensured that no additional copies of the file called ctl-privkey are placed because exposing them causes impersonation of the controller. The other generation process is the public key, as shown in Figure 4.4. Also, by using the RSA algorithm and after generation, the public key is added inside the certificates in order to also use it in the process of signing the trusted devices.

Figure 4.4: Public key of Controller

2. Creation Certificates Result

In order to generate a certificate for the controller, through writing the following commands as shown in Figure 4.5 in the same PKI in which they were created:

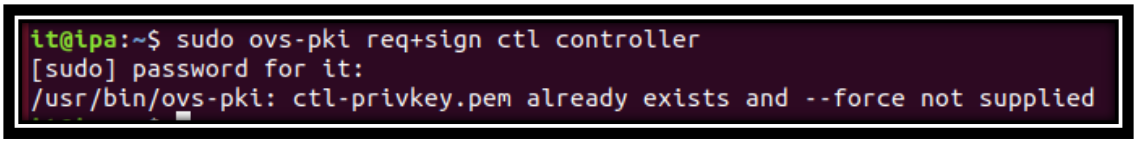


Figure 4.5: Generate Controller certificate

It is important to remain a copy of these certificates inside the controller in order to use them during execution.

4.4.2 Implementation within the Switch

After the keys have been created for the controller, the same details are then created for the switch

1. Creation Keys Result

After the public key and private key have been created inside the controller, now a public key and a private key are generated to the switch. The switch needs a key pair to prove its identity and validity to the controller so that it can receive and exchange messages and information between them. First, the private key for the switch is generated using RSA in the same way that generate the key for the controller, as shown in Figure 4.6.

Figure 4.6 : Private Key of switch

After that, the public key is generated using the RSA algorithm, as shown in Figure 4.7.

Figure 4.7: Public Key of switch

2. Creation Certificates Result

After the certificate has been created to the controller, the certificate is now created to the switch in the same way as shown in Figure 4.8.

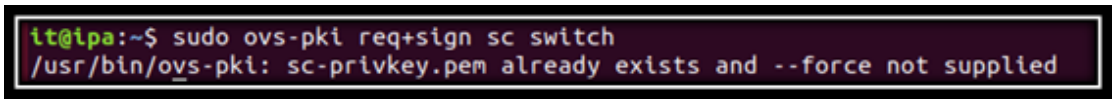
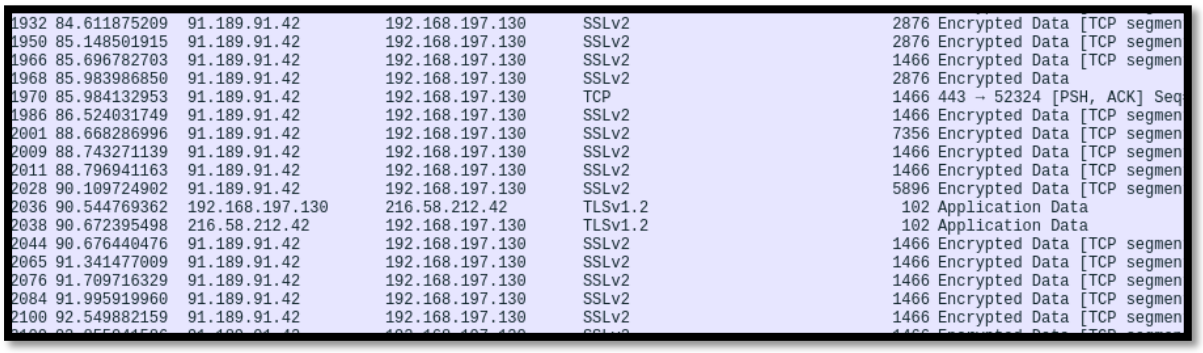


Figure 4.8 : Generating Certificate to the switch

4.5 Key Distribution Result

The process of distributing the keys begins through the process of exchanging the message between the controller and the switch; when the switch sends a message (Packet-in) to the controller, the controller first checks the device from which the packet came to ensure the authority of the device, and then the controller through flow messages, sends certificates, which it contains the keys to be sent via the SSL protocol to the devices in the data plane to be used in the process of encryption and decryption of messages.

Figure 4.9 shows a snapshot of the data traffic in Wireshark, and it is shown that the data is encrypted, which means that the keys have been used by the devices and the distribution process has been carried out correctly using the SSL protocol.

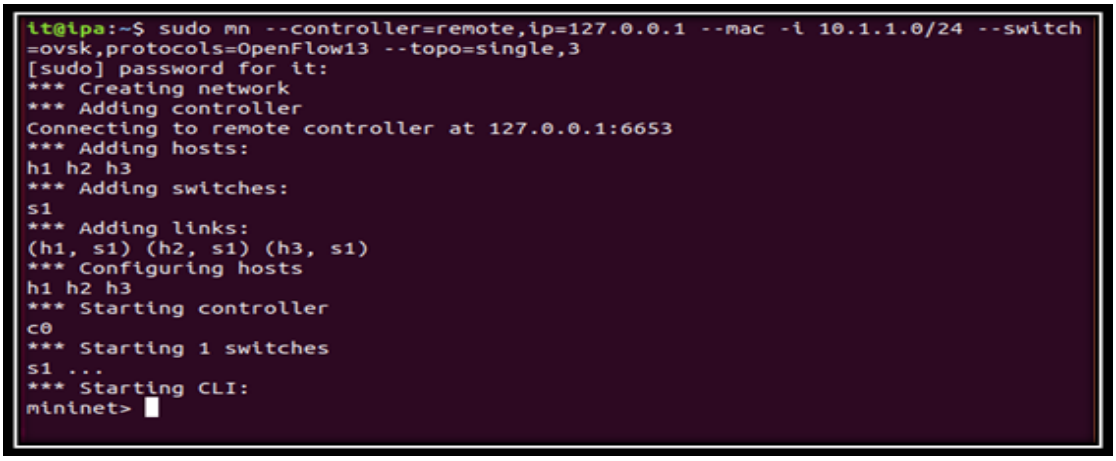


1932	84.611875209	91.189.91.42	192.168.197.130	SSLv2	2876 Encrypted Data [TCP segmen
1950	85.148501915	91.189.91.42	192.168.197.130	SSLv2	2876 Encrypted Data [TCP segmen
1966	85.696782703	91.189.91.42	192.168.197.130	SSLv2	1466 Encrypted Data [TCP segmen
1968	85.983986850	91.189.91.42	192.168.197.130	SSLv2	2876 Encrypted Data
1970	85.984132953	91.189.91.42	192.168.197.130	TCP	1466 443 → 52324 [PSH, ACK] Seq
1986	86.524031749	91.189.91.42	192.168.197.130	SSLv2	1466 Encrypted Data [TCP segmen
2001	88.668286996	91.189.91.42	192.168.197.130	SSLv2	7356 Encrypted Data [TCP segmen
2009	88.743271139	91.189.91.42	192.168.197.130	SSLv2	1466 Encrypted Data [TCP segmen
2011	88.796941163	91.189.91.42	192.168.197.130	SSLv2	1466 Encrypted Data [TCP segmen
2028	90.109724902	91.189.91.42	192.168.197.130	SSLv2	5896 Encrypted Data [TCP segmen
2036	90.544769362	192.168.197.130	216.58.212.42	TLSv1.2	102 Application Data
2038	90.672395498	216.58.212.42	192.168.197.130	TLSv1.2	102 Application Data
2044	90.676440476	91.189.91.42	192.168.197.130	SSLv2	1466 Encrypted Data [TCP segmen
2065	91.341477009	91.189.91.42	192.168.197.130	SSLv2	1466 Encrypted Data [TCP segmen
2076	91.709716329	91.189.91.42	192.168.197.130	SSLv2	1466 Encrypted Data [TCP segmen
2084	91.995919960	91.189.91.42	192.168.197.130	SSLv2	1466 Encrypted Data [TCP segmen
2100	92.549882159	91.189.91.42	192.168.197.130	SSLv2	1466 Encrypted Data [TCP segmen

Figure 4.9: Data traffic in Wireshark

4.6 Generating Network Topology Result

For each of the two measured conditions, the same number of devices and switch were chosen to monitor and compare the results. The work implementation is scalable as the user can define the number of nodes and network devices as needed. Figure 4.10 shows a snapshot of the program for choosing the type of network topology and determining the number of devices used three nodes, one switch and one controller to perform the test on the network.



```
it@ipa:~$ sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --switch
=ovsk,protocols=OpenFlow13 --topo=single,3
[sudo] password for it:
*** Creating network
*** Adding controller
Connecting to remote controller at 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Figure 4.10 : Generating Network Topology

Figure 4.11 shows the final design of the network used and the links used between them, where the controller communicates with the switch via a logical link, but between the switch and the devices, a physical link was used.

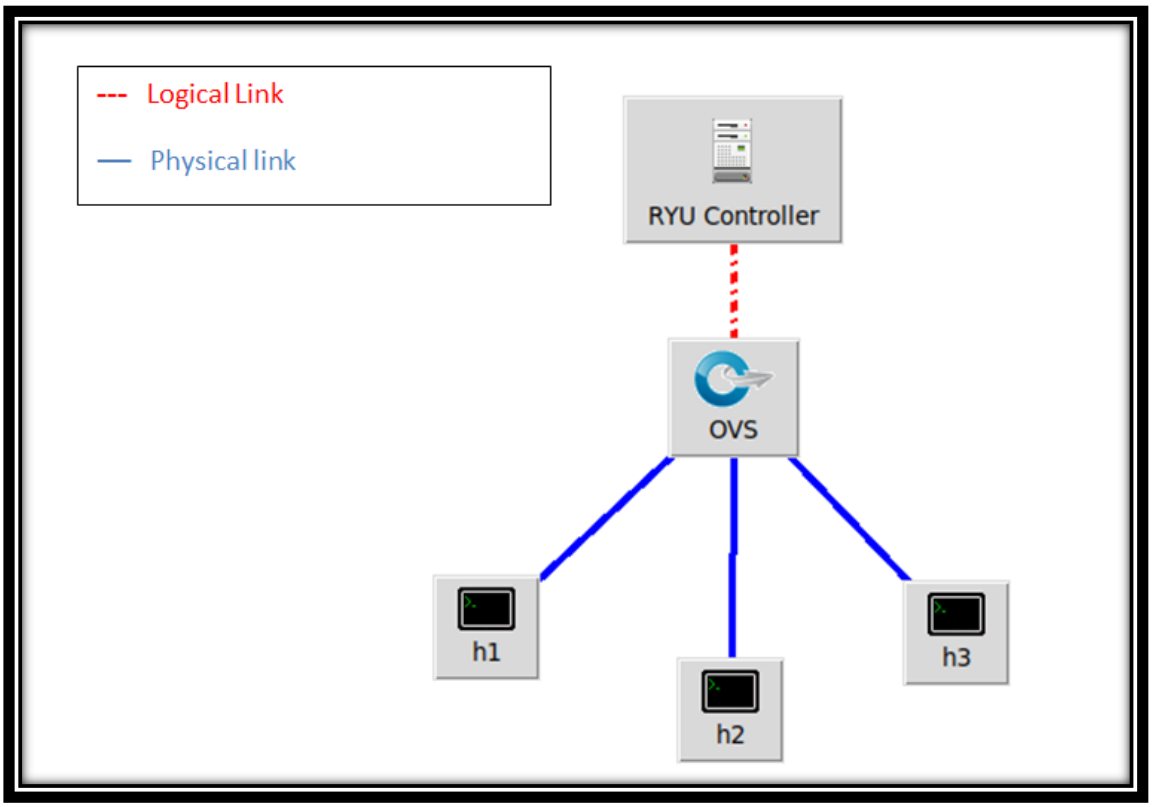


Figure 4.11 snapshot of network topology

4.7 Network Examine Result

To analyze the statistics on network performance, the network topology is implemented on one switch with three. The network topology was created with two different scenarios in which the network state differs, as shown in Table 4.5, in order to evaluate the performance and effectiveness of the network in detecting the presence of an intruder and analyze the results obtained.

Table 4.5: Experimental table of cases for a network

Cases	Number of switch	Number of nodes	State of Network
Case 1	1	3	Normal Traffic
Case 2	1	3	Intruder Traffic

Now the following steps are followed in both cases step by step to conduct the experiment on the RYU controller using the Mininet simulation:

Step 1: The first stage is the process of activating the secureswitch_v2.py program inside the controller in order to activate the intrusion detection application before creating the network in order to detect any intruder that enters the network. You must first make sure that the program is located inside the ryu folder. After that, executing the application through the following command:

```
/home/.local/lib/python2.7/sitepackages/ryu/app/secureswitch_v2.
```

Step 2: The next stage is to create the network by running the mininet single topology by providing the network topology name with the following command:

```
"sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --switch
=ovsk,protocols=OpenFlow13 --topo=single,3"
```

After executing this command, the network consisting of one switch and three devices is created. To verify the command, by check the connection of all hosts in the network by using the command:

```
mininet > pingall command
```

Step 3: In this step, the connection between the two devices is determined. It have used the connection between the first host and the second host through the command:

```
mininet > Xterm h1 h2.
```

This command will open two windows. In this step, the state of the network will be determined, either by specifying the connection as a secure connection (the first case) or specifying the device that will behave as the intruder (the second case).

Step 4: In this step, it need to generate the traffic between the two devices and then monitor the events through the iPerf tool. First, open the configuration window and write the following command:

```
iperf -s -p 5566 -i 1 > result
```

The "result" is a file in which the command result is stored. After that, it create the traffic through the window of the client; the client needs to know the address of the server. After that, it starts to connect via the command:

```
iperf -c 10.1.1.2 -p 5566 -t 15.
"15 represents the time in seconds"
```

Step 5: It is to return to the file that was created with the name "results" and displayed it by means of the command: "More results" To obtain the required information in order to produce specific results for this experiment.

Step 6: The last step is to create graphs of the obtained results using : gnuplot graphing tool.

It write the name of the tool inside the client window, and then writing the following command:

```
plot “results” title “Tcp_Flow” with line points.
```

With the same steps from 1 to step 6, all the cases mentioned in the Table 4.5 are implemented to extract and analyze the results of each case.

4.7.1 Case 1

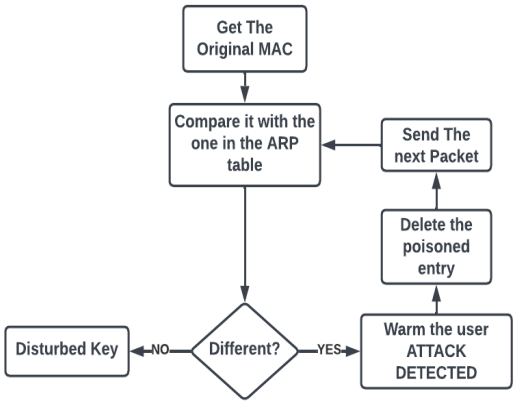
This case represents the normal network state, and by implementing the previous steps that have been explained, the client (h1) begins to initiate communication with the destination (h2). When the package is sent for the first time, the package enters two stages to

discover whether the device that sent this package is a trusted device or a fake device.

1. Using Physical MAC address

When the package entered the first detection stage, where the MAC address in the ARP request was compared with the address of the physical MAC of the device, as shown in Table 4.6.

Table 4.6 : MAC address in ARP compared with physical MAC

Before Detection	Detection process	After Detection	Result	Key Distribution
MAC in the ARP request AB:EF:12:30:50:70		The physical MAC is AB:EF:12:30:50:70 and is the same MAC in ARP request	100% Normal	Yes

The result gave a 100% similarity rate, so there is no need to enter the package in the next stage. The controller adds this host IP address and MAC in the table to facilitate the device connection directly next time, as shown in Figure 4.12.

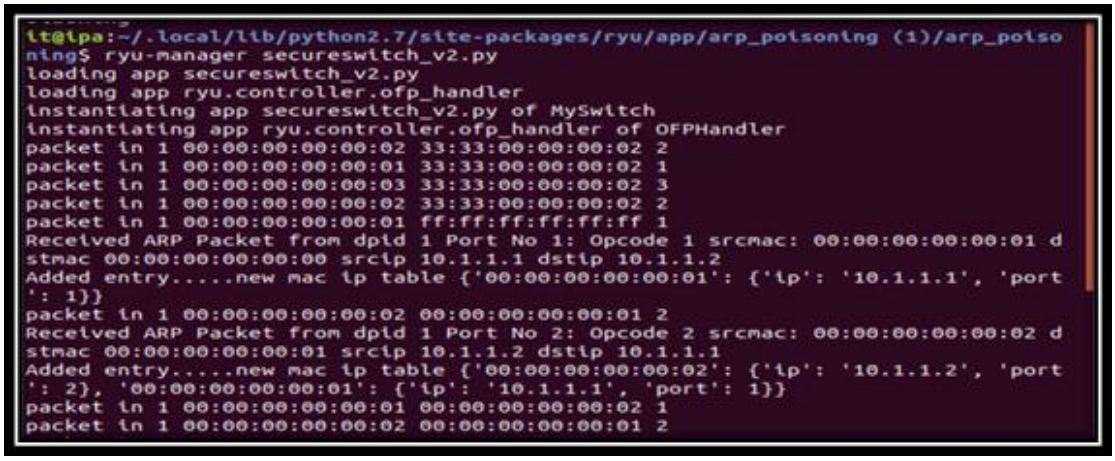


Figure 4.12 : Allow begin communication by the controller

The keys are distributed to the devices for use in encrypting and decryption messages, and then the communication process begins between the two devices, as shown in Figure 4.13.

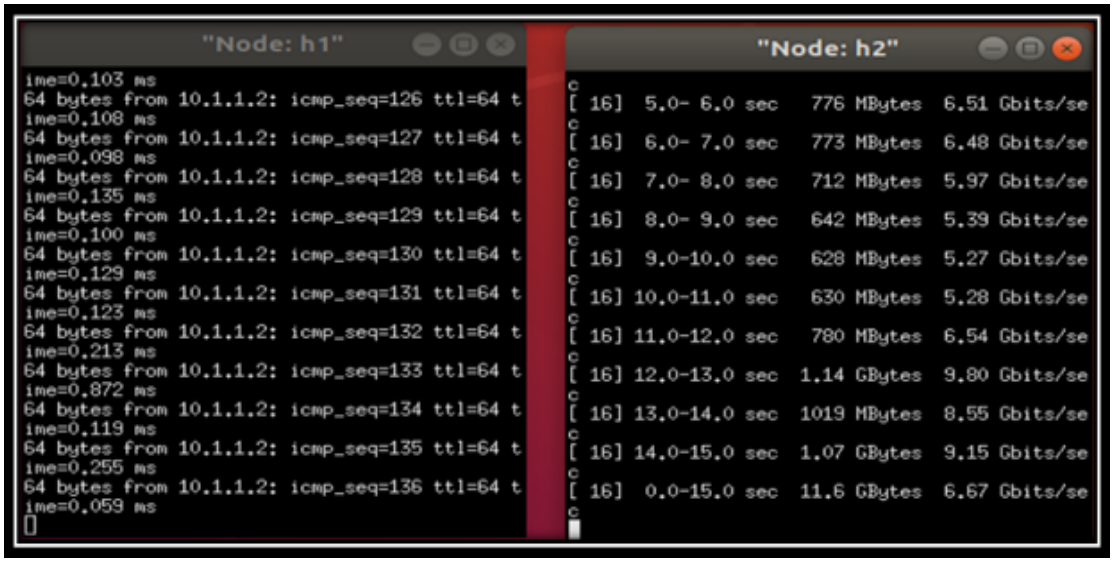


Figure 4.13: begin Communication between two host

2. Throughput Result

Figure 4.14 showed the normal TCP flow traffic and calculate the throughput of the network during the determined period of time.

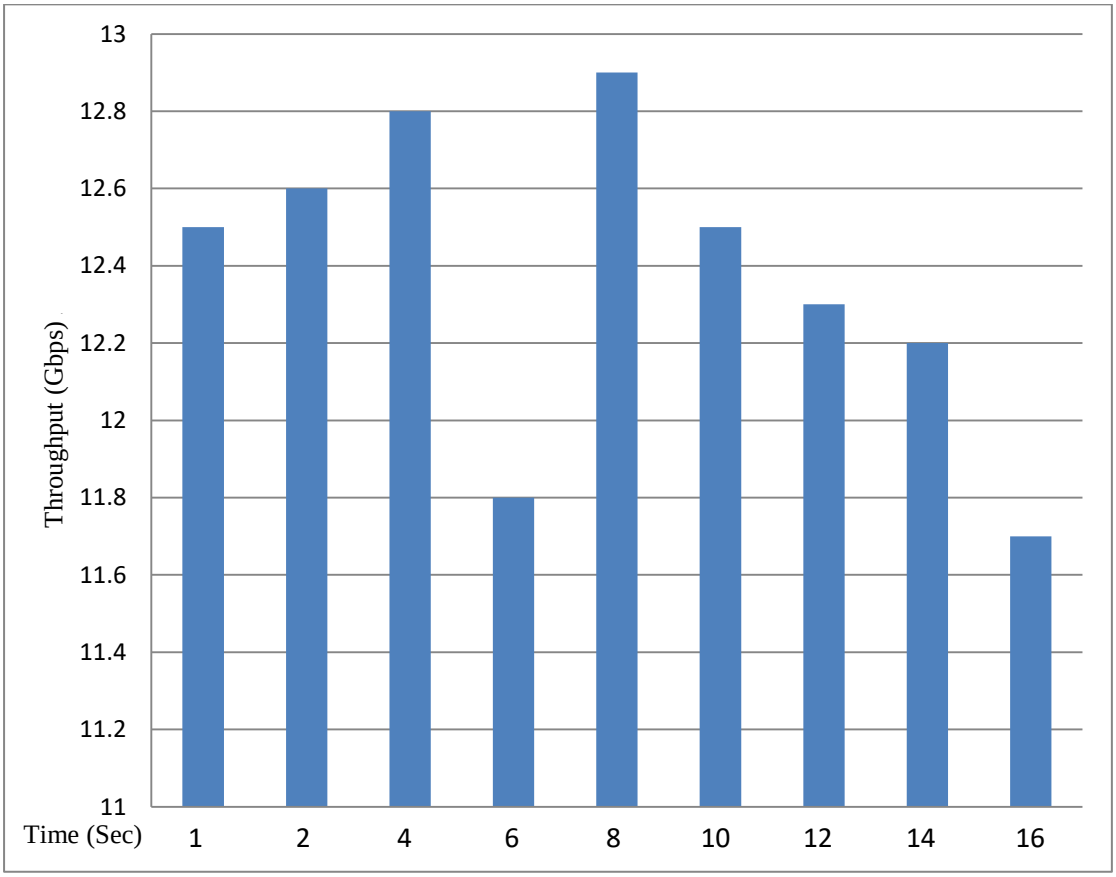


Figure 4.14 : Throughput of TCP flow in normal traffic

4.7.2 Case 2

This case represents the presence of Spoofing in the network, and also, by executing the same previous steps that were explained, the client (h3) starts trying to initiate a connection to the destination (h2). When the packet is sent for the first time, the packet goes through two stages to detect whether the device that sent this packet is a trusted device or a fake device:

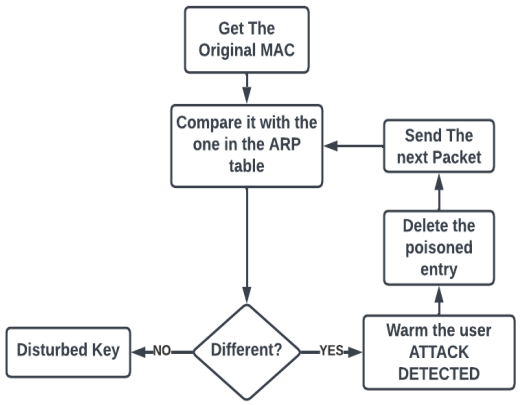
1. First Checking using Physical MAC address

The sender (h3) sends an ARP request packet to the switch to discover the destination MAC address if the sending device's ARP table does not already have it. The attacker(h1) observes this request and responds on the victim's machine (the sending machine) with a fake ARP response packet. The most recent response supersedes the oldest in the

ARP protocol. As a result, the spoof device communicates as the intended device.

To prevent such a situation from occurring, the packet is inserted in the first stage of ensuring the authority of the device.

Table 4.7: MAC Detection process.

Before Detection	Detection process	After Detection	Result	Key Distribution
MAC in the ARP request AB:EF:12:30:50:70		The physical MAC is AB:EF:9:30:22:10 And is the different from MAC in ARP request	90% Attack	No until ensure in the second detection stage

Depending on the attack detection process shown in Table 4.7, it gave a 90% rate of existence of the attacker in the network.

The Figure 4.15 show how the notification that is sent by the controller to notify the presence of an attacker and at the same time it deletes the port coming from it to permanently cut off the connection with the attacker.

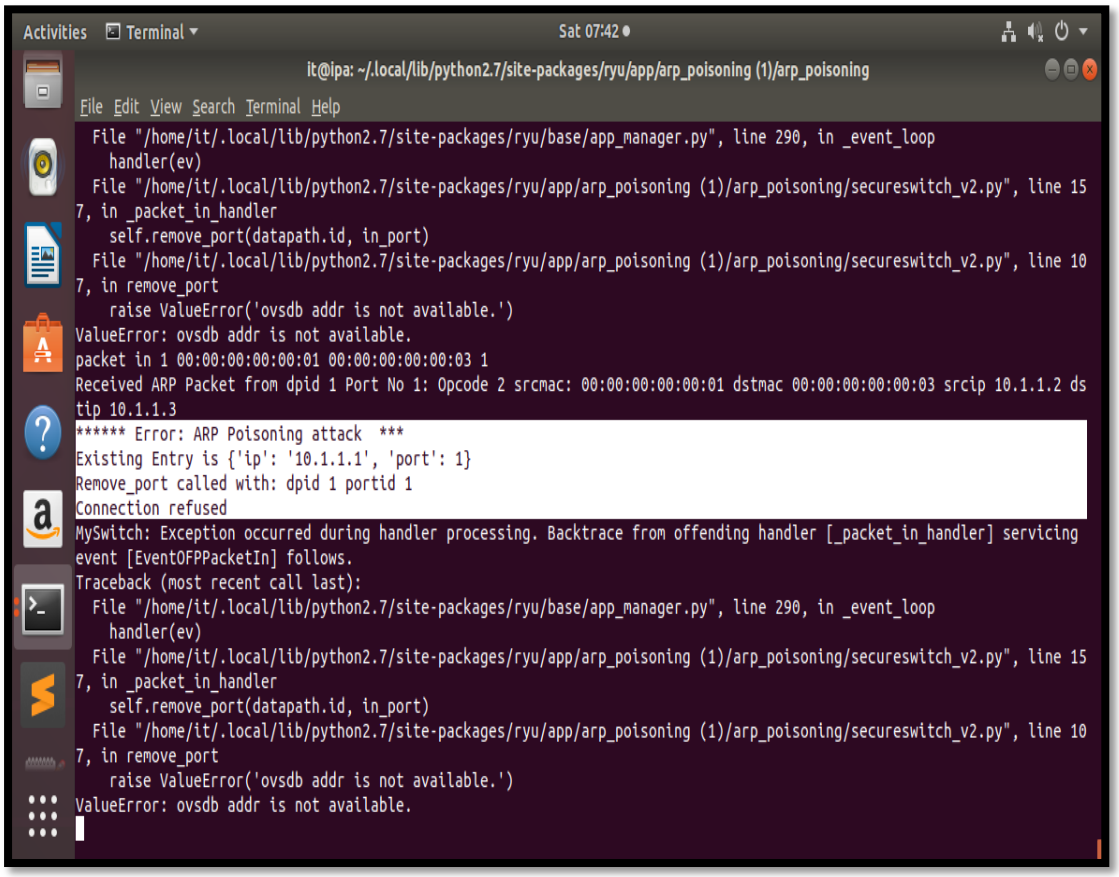
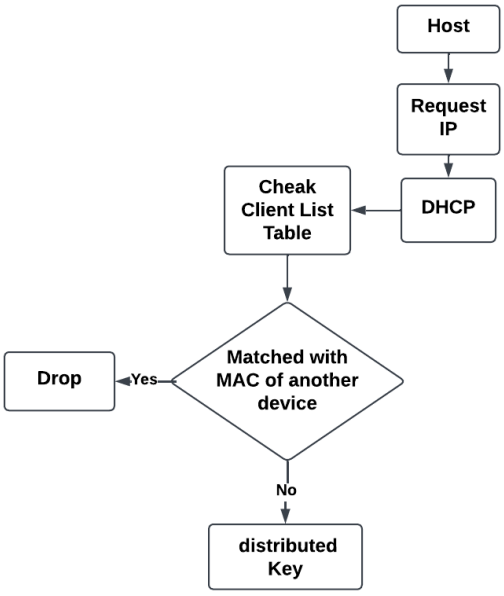


Figure 4.15: Notification sent by the controller when an intruder is detected.

2. Second Checking using DHCP server

When the device accesses the network directly, it takes the IP address through the DHCP server. After going through the first stage, which revealed 90% of the presence of an attack, the packet now enters the second stage of verification through the table list located inside the DHCP server, which contains all the original machines previously registered when the device connected to the network for the first time. According to the detection process used in Table 4.8 , it gave a result 85% rate in detecting the presence of the attacker.

Table 4.8 : detecting the presence of the attacker with DHCP.

Before Detection	Detection process	After Detection	Result	Key Distribution
MAC in the ARP request AB:EF:12:30:50:70		The MAC address in the ARP request is matched with the address of another device registered in the address list within DHCP	85% Attack	No

As shown in the Figure 4.16 the process of detecting the difference of the MAC through DHCP.

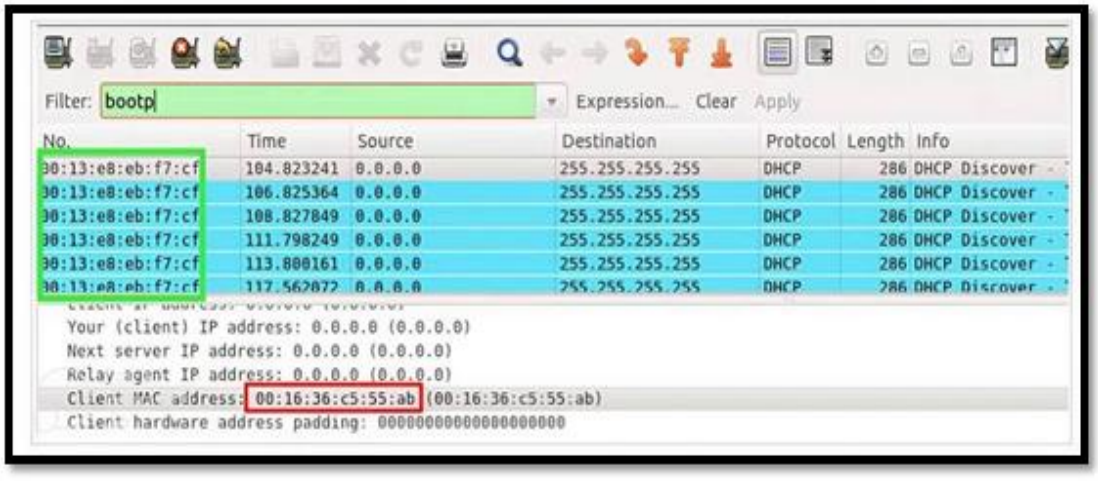


Figure 4.16 : MAC difference detection process by DHCP

Through the two results obtained in case2, the final result Which confirms the presence of the attacker can be extracted by the following equation:

Final result= $\frac{cheaking1+Cheaking2}{2} = \frac{90\%+85\%}{2} = 88\%$

3. Throughput Result

The results in Figure 4.17 showed that the proposed system enhanced throughput and made it stable without packet loss or network degradation.

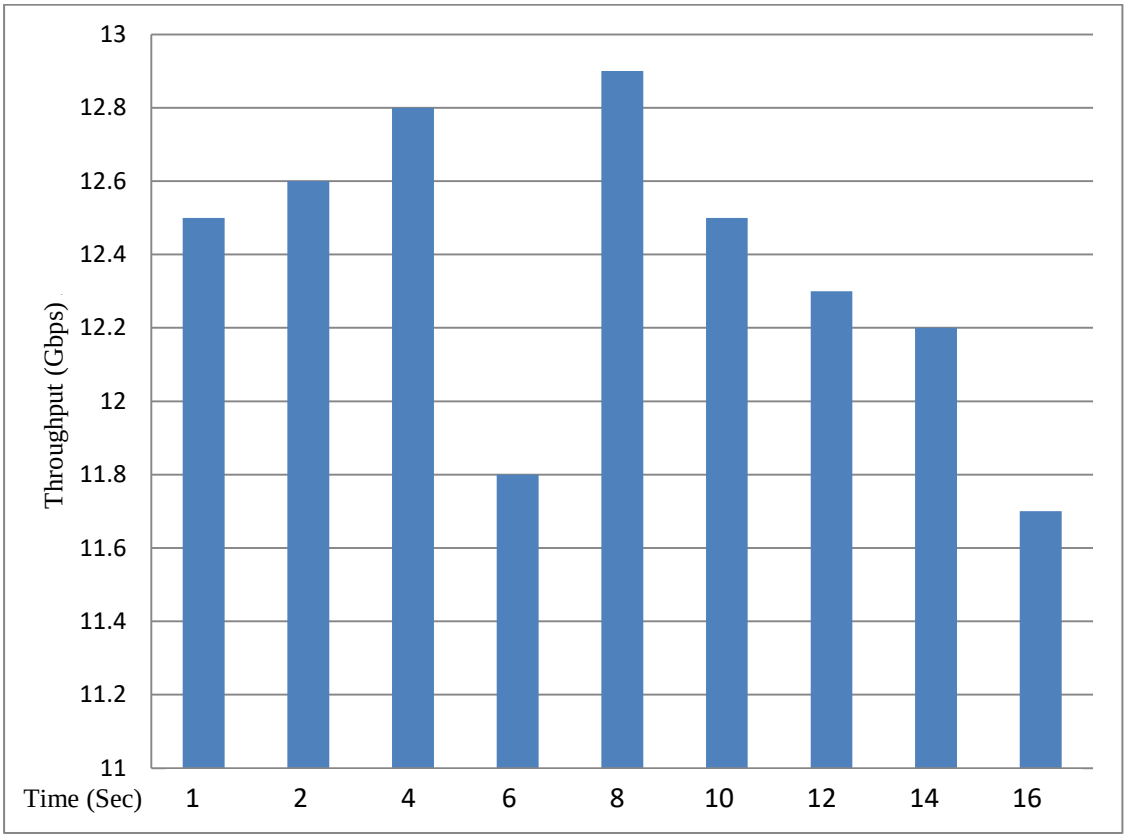


Figure 4.17 : Throughput of TCP flow after spoofing traffic.

4.8 Performance Analysis

In this section, the obtained results are discussed and analyzed, where two graphs representing the results of the cases are analyzed in Table 4.5. To evaluate network performance, throughput is the most important parameter, which will suffice our goal in this experiment. Figure 4.16 shows the results of communication between the client and the server in the normal state, while Figure 4.18 represents the results of the network when there is a case of intrusion. In comparison between the two results, it note that there is no change in the throughput because the

program that was implemented inside the controller prevented the fake device from entering the network and preventing its communication with the rest of the computers, so no change appeared on the network performance.

4.8.1 Throughput analyzer with 50 nodes

Figure 4.18 showed the network creation with 50 total number of nodes in Mininet simulation tool.

```
it@ipa:~$ sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --switc
h=ovsk,protocols=OpenFlow13 --topo=single,50
[sudo] password for it:
Sorry, try again.
[sudo] password for it:
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22
h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42
 h43 h44 h45 h46 h47 h48 h49 h50
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9, s1
) (h10, s1) (h11, s1) (h12, s1) (h13, s1) (h14, s1) (h15, s1) (h16, s1) (h17, s
1) (h18, s1) (h19, s1) (h20, s1) (h21, s1) (h22, s1) (h23, s1) (h24, s1) (h25,
s1) (h26, s1) (h27, s1) (h28, s1) (h29, s1) (h30, s1) (h31, s1) (h32, s1) (h33,
 s1) (h34, s1) (h35, s1) (h36, s1) (h37, s1) (h38, s1) (h39, s1) (h40, s1) (h41
, s1) (h42, s1) (h43, s1) (h44, s1) (h45, s1) (h46, s1) (h47, s1) (h48, s1) (h4
9, s1) (h50, s1)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22
h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42
 h43 h44 h45 h46 h47 h48 h49 h50
*** Starting controller
```

Figure 4.18: Network creation of 50 nodes in Mininet simulation.

Figure 4.19 showed the results obtained by performing the transfer between client and server with the power of nodes supported by the network limited to 50 nodes. It is observed from the graph that the average throughput remains at 1.65 Gbps. It also showed that the variations are very high over the simulation duration of 100 s.

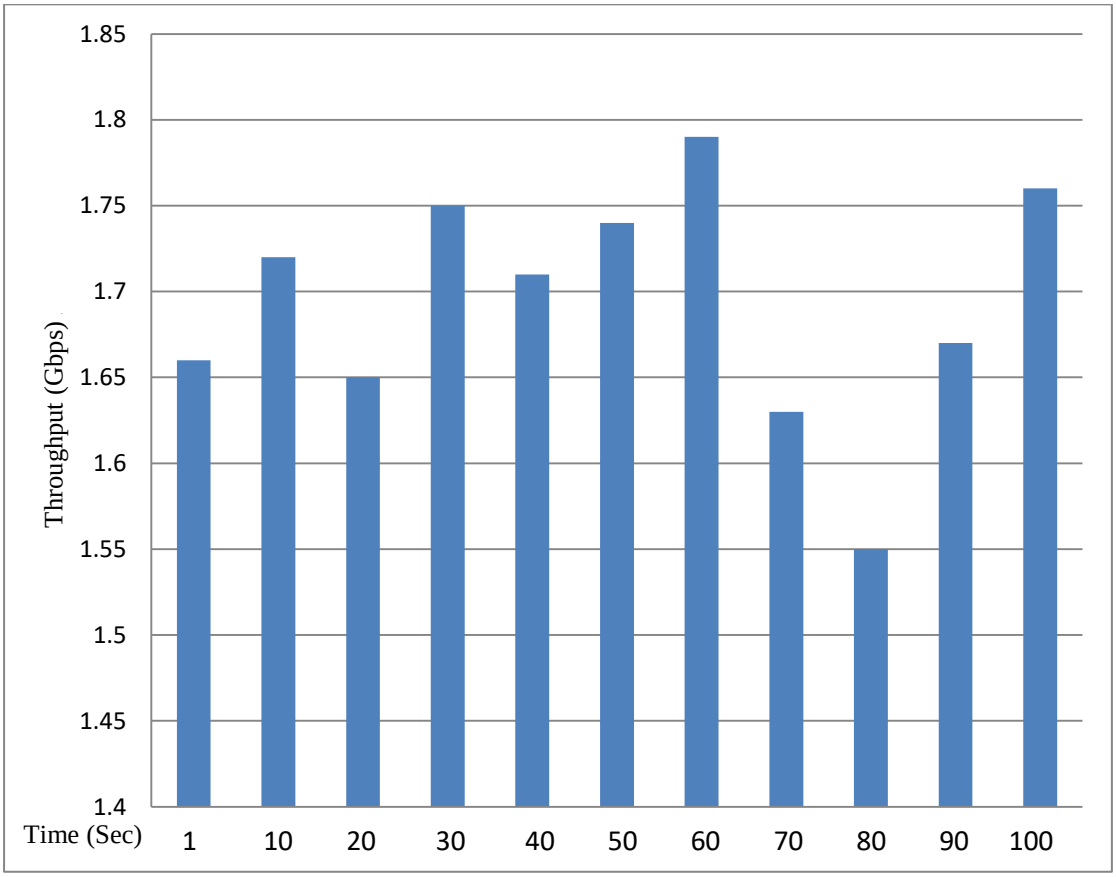


Figure 4.19: Throughput of TCP flow of 50 nodes.

4.8.2 Throughput analyzer with 100 nodes

Figure 4.20 showed the network creation with 100 nodes with the remote controller to send and received packets.

```
it@ipa:~$ sudo mn --controller=remote,ip=127.0.0.1 --mac -i 10.1.1.0/24 --switc
h=ovsk,protocols=OpenFlow13 --topo=single,100
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10 h11 h12 h13 h14 h15 h16 h17 h18 h19 h20 h21 h22
h23 h24 h25 h26 h27 h28 h29 h30 h31 h32 h33 h34 h35 h36 h37 h38 h39 h40 h41 h42
 h43 h44 h45 h46 h47 h48 h49 h50 h51 h52 h53 h54 h55 h56 h57 h58 h59 h60 h61 h6
2 h63 h64 h65 h66 h67 h68 h69 h70 h71 h72 h73 h74 h75 h76 h77 h78 h79 h80 h81 h
82 h83 h84 h85 h86 h87 h88 h89 h90 h91 h92 h93 h94 h95 h96 h97 h98 h99 h100
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1) (h5, s1) (h6, s1) (h7, s1) (h8, s1) (h9, s1
) (h10, s1) (h11, s1) (h12, s1) (h13, s1) (h14, s1) (h15, s1) (h16, s1) (h17, s
1) (h18, s1) (h19, s1) (h20, s1) (h21, s1) (h22, s1) (h23, s1) (h24, s1) (h25,
s1) (h26, s1) (h27, s1) (h28, s1) (h29, s1) (h30, s1) (h31, s1) (h32, s1) (h33,
 s1) (h34, s1) (h35, s1) (h36, s1) (h37, s1) (h38, s1) (h39, s1) (h40, s1) (h41
, s1) (h42, s1) (h43, s1) (h44, s1) (h45, s1) (h46, s1) (h47, s1) (h48, s1) (h4
9, s1) (h50, s1) (h51, s1) (h52, s1) (h53, s1) (h54, s1) (h55, s1) (h56, s1) (h
57, s1) (h58, s1) (h59, s1) (h60, s1) (h61, s1) (h62, s1) (h63, s1) (h64, s1) (
h65, s1) (h66, s1) (h67, s1) (h68, s1) (h69, s1) (h70, s1) (h71, s1) (h72, s1)
(h73, s1) (h74, s1) (h75, s1) (h76, s1) (h77, s1) (h78, s1) (h79, s1) (h80, s1)
 (h81, s1) (h82, s1) (h83, s1) (h84, s1) (h85, s1) (h86, s1) (h87, s1) (h88, s1
) (h89, s1) (h90, s1) (h91, s1) (h92, s1) (h93, s1) (h94, s1) (h95, s1) (h96, s
```

Figure 4.20: Network creation of 100 nodes in Mininet simulation.

Likewise, stability is a major concern Figure 4.21 which shows the throughput graph for the scenario with 100 nodes. It is even worse in terms of stability compared to Figure 4.19 of 50 nodes.

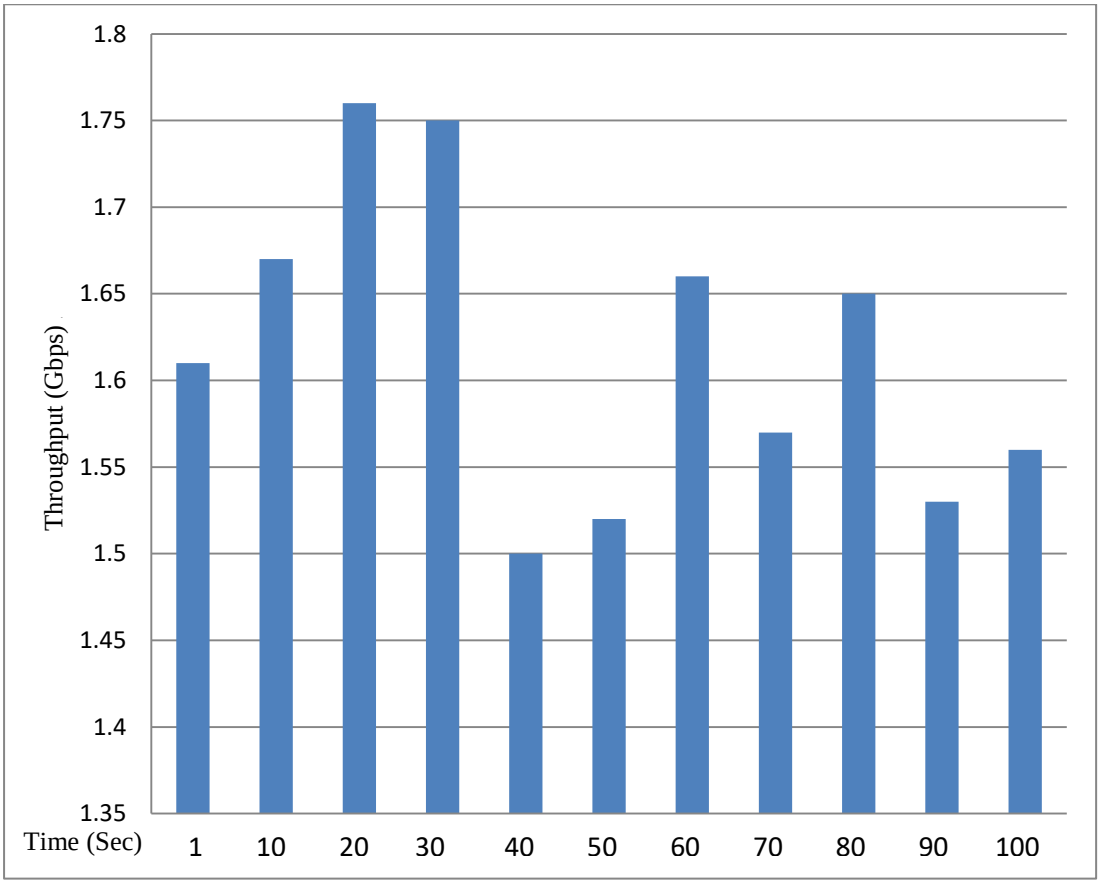


Figure 4.21: Throughput of TCP flow of 100 nodes.

4.8.3 Throughput analyzer with 150 nodes

Figure 4.22 showed the network creation with 150 nodes which connect with RYU controller in SDN environment. It showed the network connection with hosts with remote controller and switch to pass packets among network elements.

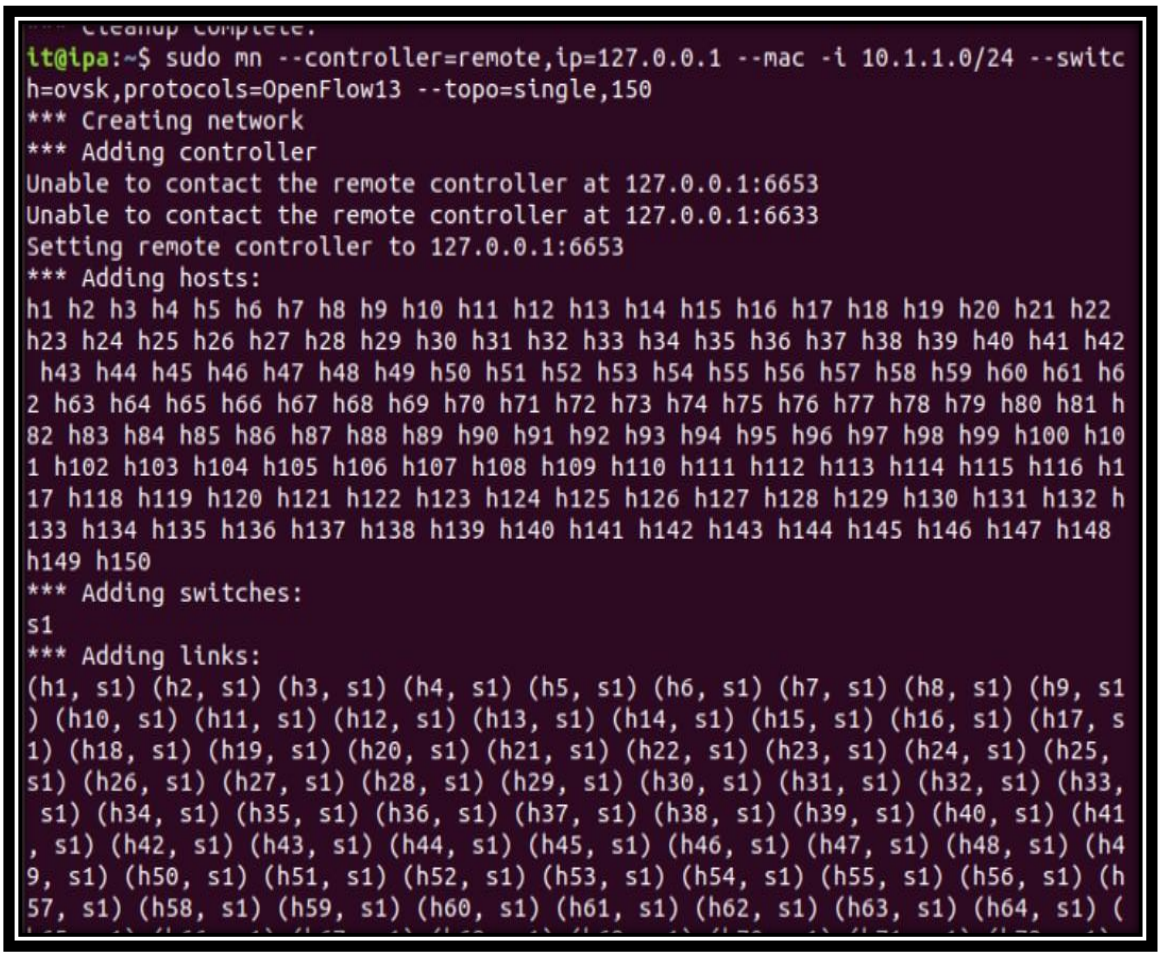


Figure 4.22: Network creation of 150 nodes in Mininet simulation.

Again, if observed well, Figure 4.23 is somewhat stable even with a number of nodes equal to 150. However, there are few instances of fluctuation in the behavior of the network . The simulation was running well, but towards the end of the simulation time, slow conditions were frequently observed which degraded network performance . All case studies of 50, 100 and 150 nodes is applied with RYU resource-intensive controller and utilizes CPU and RAM optimally, it showed the degrading performance under an increasing number of nodes, as network throughput is decreased.

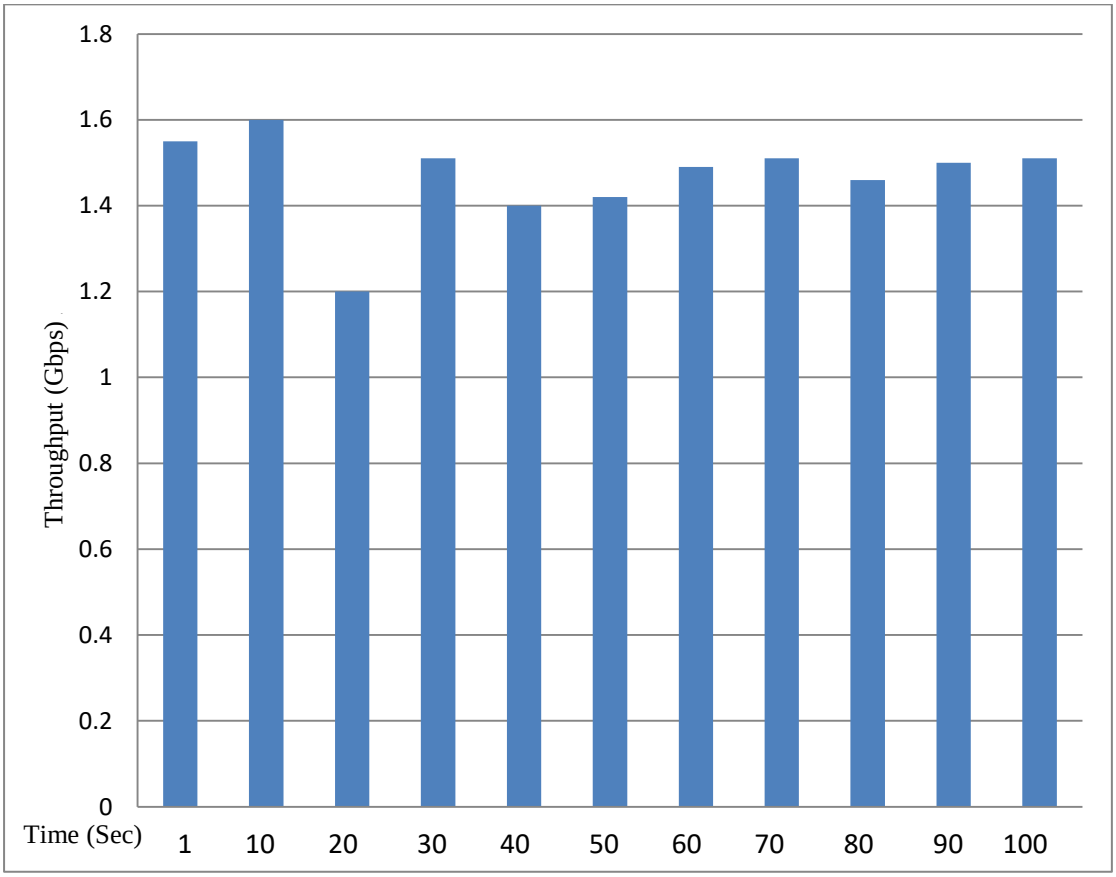


Figure 4.23: Throughput of TCP flow of 150 nodes.

Chapter Five

Conclusions and Suggestions for Future Works

5.1 Conclusions

This chapter explains the proposed system conclusions and the main suggestions for future works they can be summarized as :

1- The proposed SDN network is established consisting of one RYU controller, one openFlow virtual switch, and three devices for the to transfer secure data and prevent malicious devices.

2- The proposed system mechanism is based on distributing keys through certificates, managing those keys automatically in a public key infrastructure, and sending via a secured SSL channel to devices whose authorization has been proven through the DHCP server and also through the ARP MAC address and by using the RSA algorithm to encrypt the connection.

3- The proposed system results showed that the total average of throughput of TCP flow in normal traffic and the case of presence of spoofing attack in the network is 12.366 Gbps and total average of time is 8.111 seconds which showed the network performance is stable and without loss rate.

4- The proposed system tested with total average required time is 50.090 seconds and different number of nodes cases as :

A- Throughput of TCP flow of 50 nodes.

The total average of throughput is 1.693 Gbps.

B- Throughput of TCP flow of 100 nodes.

The total average of throughput is 1.616 Gbps.

C- Throughput of TCP flow of 150 nodes.

The total average of throughput is 1.468 Gbps.

5- The results showed that throughput is decreased due to increased number of nodes and required time to complete processes between controller and nodes

5.2 Suggestions for Future Works

The following guidelines can be used to realize a number of considerations for the future expansion of the current research:

- Making a development of key management by Appling symmetric key management.
- Using machine learning to distinguish between normal traffic and malicious traffic.
- Creating of the interface between application and controller and adding some instructions in the application plane, for example, adding or deleting some peers, and also specifying a period of time for the session to use keys.
- Use the Internet of Things (IoT) as a data plane instead of hardware to improve performance in network with high numbers of device, provide more direct control over routing, analyze network traffic and manage the network effectively.

REFERENCES

[1] Sicari, S., Rizzardi, A., & Coen-Porisini, A. (2020). 5G In the internet of things era: An overview on security and privacy challenges. *Computer Networks*, *179*, 107345.

[2] Waseem, Q., Alshamrani, S. S., Nisar, K., Wan Din, W. I. S., & Alghamdi, A. S. (2021). Future technology: Software-defined network (SDN) forensic. *Symmetry*, *13*(5), 767.

[3] Keshari, S. K., Kansal, V., & Kumar, S. (2021). A systematic review of quality of services (QoS) in software defined networking (SDN). *Wireless Personal Communications*, *116*, 2593-2614.

[4] Alzahrani, A. O., & Alenazi, M. J. (2021). Designing a network intrusion detection system based on machine learning for software defined networks. *Future Internet*, *13*(5), 111.

[5] Onyema, E. M., Kumar, M. A., Balasubaramanian, S., Bharany, S., Rehman, A. U., Eldin, E. T., & Shafiq, M. (2022). A security policy protocol for detection and prevention of internet control message protocol attacks in software defined networks. *Sustainability*, *14*(19), 11950.

[6] Yungaicela-Naula, N. M., Vargas-Rosales, C., Pérez-Díaz, J. A., & Zareei, M. (2022). Towards security automation in software defined networks. *Computer Communications*, *183*, 64-82.

[7] Gupta, N., Maashi, M. S., Tanwar, S., Badotra, S., Aljebreen, M., & Bharany, S. (2022). A comparative study of software defined networking controllers using mininet. *Electronics*, *11*(17), 2715.

[8] Yigit, B., Gur, G., Tellenbach, B., & Alagoz, F. (2019). Secured communication channels in software-defined networks. *IEEE Communications Magazine*, *57*(10), 63-69.

[9] Karmakar, K. (2019). Techniques for securing software defined networks and services (Doctoral dissertation, dissertation, University of Newcastle).

[10] Ali, A., & Yousaf, M. M. (2020). Novel three-tier intrusion detection and prevention system in software defined network. IEEE Access, 8, 109662-109676.

[11] Hu, T., Zhang, Z., Yi, P., Liang, D., Li, Z., Ren, Q., ... & Lan, J. (2021). SEAPP: A secure application management framework based on REST API access control in SDN-enabled cloud environment. Journal of Parallel and Distributed Computing, 147, 108-123.

[12] Lowery, C. J. (2021). INFERRING NETWORKING EVENTS FROM TRANSPORT LAYER SECURITY–ENCRYPTED TRAFFIC (Doctoral dissertation, dissertation, Monterey, CA; Naval Postgraduate School).

[13] Ghaly, S., & Abdullah, M. Z. (2021). Design and implementation of a secured SDN system based on hybrid encrypted algorithms. TELKOMNIKA (Telecommunication Computing Electronics and Control), 19(4), 1118-1125.

[14] Jafarian, T., Masdari, M., Ghaffari, A., & Majidzadeh, K. (2021). A survey and classification of the security anomaly detection mechanisms in software defined networks. *Cluster Computing*, *24*, 1235-1253.

[15] Morsy, S. M., & Nashat, D. (2022). D-ARP: An Efficient Scheme to Detect and Prevent ARP Spoofing. *IEEE Access*, *10*, 49142-49153.

[16] Paolucci, F., Cugini, F., Castoldi, P., & Osiński, T. (2021). Enhancing 5G SDN/NFV edge with P4 data plane programmability. *IEEE Network*, *35*(3), 154-160.

[17] Almadani, B., Beg, A., & Mahmoud, A. (2021). DSF: A distributed sdn control plane framework for the east/west interface. *IEEE Access*, *9*, 26735-26754.

[18] Manzanares-Lopez, P., Muñoz-Gea, J. P., & Malgosa-Sanahuja, J. (2021). Passive in-band network telemetry systems: The potential of programmable data plane on network-wide telemetry. *IEEE Access*, *9*, 20391-20409.

[19] Keshari, S. K., Kansal, V., & Kumar, S. (2021). A systematic review of quality of services (QoS) in software defined networking (SDN). *Wireless Personal Communications*, *116*, 2593-2614.

[20] Islam, M. J., Rahman, A., Kabir, S., Karim, M. R., Acharjee, U. K., Nasir, M. K., ... & Wu, S. (2021). Blockchain-SDN-based energy-aware and distributed secure architecture for IoT in smart cities. *IEEE Internet of Things Journal*, *9*(5), 3850-3864.

[21] Do, H. M., Gregory, M. A., & Li, S. (2021). SDN-based wireless mobile backhaul architecture: Review and challenges. *Journal of Network and Computer Applications*, *189*, 103138.

[22] Zhang, Y., & Chen, M. (2022). Performance evaluation of Software-Defined Network (SDN) controllers using Dijkstra's algorithm. *Wireless Networks*, *28*(8), 3787-3800.

[23] Osiński, T., Palimąka, J., Kossakowski, M., Tran, F. D., Bonfoh, E. F., & Tarasiuk, H. (2022, November). A novel programmable software datapath for software-defined networking. In *Proceedings of the 18th International*

Conference on emerging Networking EXperiments and Technologies (pp. 245-260).

[24] Sallam, A., Refaey, A., & Shami, A. (2019). On the security of SDN: A completed secure and scalable framework using the software-defined perimeter. *IEEE access*, *7*, 146577-146587.

[25] Tajiki, M. M., Shojafar, M., Akbari, B., Salsano, S., Conti, M., & Singhal, M. (2019). Joint failure recovery, fault prevention, and energy-efficient resource management for real-time SFC in fog-supported SDN. *Computer Networks*, *162*, 106850.

[26] Swami, R., Dave, M., & Ranga, V. (2019). Software-defined networking-based DDoS defense mechanisms. *ACM Computing Surveys (CSUR)*, *52*(2), 1-36.

[27] Amiri, E., Alizadeh, E., & Raeisi, K. (2019, February). An efficient hierarchical distributed SDN controller model. In *2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)* (pp. 553-557). IEEE.

[28] Latah, M., & Toker, L. (2019). Artificial intelligence enabled software-defined networking: a comprehensive overview. *IET networks*, *8*(2), 79-99.

[29] Al-Hubaishi, M., Çeken, C., & Al-Shaikhli, A. (2019). A novel energy-aware routing mechanism for SDN-enabled WSAN. *International Journal of Communication Systems*, *32*(17), e3724.

[30] Cicioğlu, M., & Çalhan, A. (2019). HUBsFLOW: A novel interface protocol for SDN-enabled WBANs. *Computer Networks*, *160*, 105-117.

[31] Derhab, A., Guerroumi, M., Gumaei, A., Maglaras, L., Ferrag, M. A., Mukherjee, M., & Khan, F. A. (2019). Blockchain and random subspace

learning-based IDS for SDN-enabled industrial IoT security. *Sensors*, *19*(14), 3119.

[32] Zwane, S., Tarwireyi, P., & Adigun, M. (2019, November). A Flow-based IDS for SDN-enabled Tactical Networks. In *2019 International Multidisciplinary Information Technology and Engineering Conference (IMITEC)* (pp. 1-6). IEEE.

[33] Pisharody, S., Natarajan, J., Chowdhary, A., Alshalan, A., & Huang, D. (2017). Brew: A security policy analysis framework for distributed SDN-based cloud environments. *IEEE transactions on dependable and secure computing*, *16*(6), 1011-1025.

[34] Carvalho, R. N., Bordim, J. L., & Alchieri, E. A. P. (2019, May). Entropy-based DoS attack identification in SDN. In *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* (pp. 627-634). IEEE.

[35] Rahouti, M., Xiong, K., Xin, Y., Jagatheesaperumal, S. K., Ayyash, M., & Shaheed, M. (2022). SDN security review: Threat taxonomy, implications, and open challenges. *IEEE Access*, *10*, 45820-45854.

[36] Javeed, D., Gao, T., & Khan, M. T. (2021). SDN-enabled hybrid DL-driven framework for the detection of emerging cyber threats in IoT. *Electronics*, *10*(8), 918.

[37] Jmal, R., Ghabri, W., Guesmi, R., Alshammari, B. M., Alshammari, A. S., & Alsaif, H. (2023). Distributed Blockchain-SDN Secure IoT System Based on ANN to Mitigate DDoS Attacks. *Applied Sciences*, *13*(8), 4953.

[38] Hu, T., Guo, Z., Yi, P., Baker, T., & Lan, J. (2018). Multi-controller based software-defined networking: A survey. *IEEE access*, *6*, 15980-15996.

[39] Ramprasath, J., & Seethalakshmi, V. (2021). Mitigation of malicious flooding in software defined networks using dynamic access control list. *Wireless Personal Communications*, *121*(1), 107-125.

[40] Cheng, H., Liu, J., Mao, J., Wang, M., Chen, J., & Bian, J. (2018). A compatible openflow platform for enabling security enhancement in SDN. *Security and Communication Networks*, *2018*, 1-20.

[41] Oudin, R., Antichi, G., Rotsos, C., Moore, A. W., & Uhlig, S. (2018). OFLOPS-SUME and the art of switch characterization. *IEEE Journal on Selected Areas in Communications*, *36*(12), 2612-2620.

[42] Shafiq, H., Rehman, R. A., & Kim, B. S. (2018). Services and security threats in sdn based vanets: A survey. *Wireless Communications and Mobile Computing*, *2018*.

[43] Iqbal, M., Iqbal, F., Mohsin, F., Rizwan, M., & Ahmad, F. (2019). Security issues in software defined networking (SDN): risks, challenges and potential solutions. *International Journal of Advanced Computer Science and Applications*, *10*(10).

[44] Liu, Y., Zhao, B., Zhao, P., Fan, P., & Liu, H. (2019). A survey: Typical security issues of software-defined networking. *China Communications*, *16*(7), 13-31.

[45] Kumar, R., & Goyal, R. (2019). On cloud security requirements, threats, vulnerabilities and countermeasures: A survey. *Computer Science Review*, *33*, 1-48.

[46] Eom, T., Hong, J. B., An, S., Park, J. S., & Kim, D. S. (2019). A systematic approach to threat modeling and security analysis for software defined networking. *Ieee Access*, *7*, 137432-137445.

[47] Swami, R., Dave, M., & Ranga, V. (2019). Software-defined networking-based DDoS defense mechanisms. *ACM Computing Surveys (CSUR)*, *52*(2), 1-36.

[48] Guo, X., Lin, H., Li, Z., & Peng, M. (2019). Deep-reinforcement-learning-based QoS-aware secure routing for SDN-IoT. *IEEE Internet of things journal*, *7*(7), 6242-6251.

[49] Jaballah, W. B., Conti, M., & Lal, C. (2019). A survey on software-defined VANETs: benefits, challenges, and future directions. *arXiv preprint arXiv:1904.04577*.

[50] Albahar, M. A. (2019). Recurrent neural network model based on a new regularization technique for real-time intrusion detection in SDN environments. *Security and Communication Networks*, *2019*, 1-9.

[51] Hasnat, M. A., Rumee, S. T. A., Razzaque, M. A., & Mamun-Or-Rashid, M. (2019, February). Security study of 5G heterogeneous network: current solutions, limitations & future direction. In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)* (pp. 1-4). IEEE.

[52] Al Hayajneh, A., Bhuiyan, M. Z. A., & McAndrew, I. (2020). Improving internet of things (IoT) security with software-defined networking (SDN). *Computers*, *9*(1), 8.

[53] Malik, J., Akhunzada, A., Bibi, I., Imran, M., Musaddiq, A., & Kim, S. W. (2020). Hybrid deep learning: An efficient reconnaissance and surveillance detection mechanism in SDN. *IEEE Access*, *8*, 134695-134706.

[54] Yazdinejad, A., Parizi, R. M., Dehghantanha, A., Zhang, Q., & Choo, K. K. R. (2020). An energy-efficient SDN controller architecture for IoT networks

with blockchain-based security. *IEEE Transactions on Services Computing*, *13*(4), 625-638.

[55] Han, T., Jan, S. R. U., Tan, Z., Usman, M., Jan, M. A., Khan, R., & Xu, Y. (2020). A comprehensive survey of security threats and their mitigation techniques for next-generation SDN controllers. *Concurrency and Computation: Practice and Experience*, *32*(16), e5300.

[56] Liaqat, S., Akhunzada, A., Shaikh, F. S., Giannetsos, A., & Jan, M. A. (2020). SDN orchestration to combat evolving cyber threats in Internet of Medical Things (IoMT). *Computer Communications*, *160*, 697-705.

[57] Sahoo, K. S., Tripathy, B. K., Naik, K., Ramasubbareddy, S., Balusamy, B., Khari, M., & Burgos, D. (2020). An evolutionary SVM model for DDOS attack detection in software defined networks. *IEEE access*, *8*, 132502-132513.

[58] Singh, J., & Behal, S. (2020). Detection and mitigation of DDoS attacks in SDN: A comprehensive review, research challenges and future directions. *Computer Science Review*, *37*, 100279.

[59] Hussein, A., Chadad, L., Adalian, N., Chehab, A., Elhajj, I. H., & Kayssi, A. (2020). Software-Defined Networking (SDN): the security review. *Journal of Cyber Security Technology*, *4*(1), 1-66.

[60] Isyaku, B., Mohd Zahid, M. S., Bte Kamat, M., Abu Bakar, K., & Ghaleb, F. A. (2020). Software defined networking flow table management of openflow switches performance and security challenges: A survey. *Future Internet*, *12*(9), 147.

[61] Bholebawa, I. Z., & Dalal, U. D. (2018). Performance analysis of SDN/OpenFlow controllers: POX versus floodlight. *Wireless Personal Communications*, *98*, 1679-1699.

[62] Karakus, M., & Durresi, A. (2017). Quality of service (QoS) in software defined networking (SDN): A survey. *Journal of Network and Computer Applications*, *80*, 200-218.

[63] Pourvahab, M., & Ekbatanifard, G. (2019). Digital forensics architecture for evidence collection and provenance preservation in iaas cloud environment using sdn and blockchain technology. *IEEE Access*, *7*, 153349-153364.

[64] Xie, L., Ding, Y., Yang, H., & Wang, X. (2019). Blockchain-based secure and trustworthy Internet of Things in SDN-enabled 5G-VANETs. *IEEE Access*, *7*, 56656-56666.

[65] Garg, S., Kaur, K., Kaddoum, G., Ahmed, S. H., & Jayakody, D. N. K. (2019). SDN-based secure and privacy-preserving scheme for vehicular networks: A 5G perspective. *IEEE Trnsactions on Vehicular Technology*, *68*(9), 8421-8434.

[66] Košťál, K., Bencel, R., Ries, M., Trúchly, P., & Kotuliak, I. (2019). High performance SDN WLAN architecture. *Sensors*, *19*(8), 1880.

[67] Jiasi, W., Jian, W., Jia-Nan, L., & Yue, Z. (2019). Secure software-defined networking based on blockchain. *arXiv preprint arXiv:1906.04342*.

[68] Lopez-Millan, G., Marin-Lopez, R., & Pereniguez-Garcia, F. (2019). Towards a standard SDN-based IPsec management framework. *Computer Standards & Interfaces*, *66*, 103357.

[69] Wang, C., Zhang, Y., Chen, X., Liang, K., & Wang, Z. (2019). SDN-based handover authentication scheme for mobile edge computing in cyber-physical systems. *IEEE Internet of Things Journal*, *6*(5), 8692-8701.

[70] Jimson, E. R., Nisar, K., & Hijazi, M. H. A. (2019). The state of the art of software defined networking (SDN) issues in current network architecture and a

solution for network management using the SDN. *International Journal of Technology Diffusion (IJTD)*, *10*(3), 33-48.

[71] Taurshia, A., Kathrine, G. J. W., Souri, A., Vinodh, S. E., Vimal, S., Li, K. C., & Ilango, S. S. (2022). Software-defined network aided lightweight group key management for resource-constrained Internet of Things devices. *Sustainable Computing: Informatics and Systems*, *36*, 100807.

[72] Algaradi, T. S., & Rama, B. (2022). An authenticated key management scheme for securing big data environment. *International Journal of Electrical and Computer Engineering*, *12*(3), 3238.

[73] Assafra, K., Alaya, B., & Abid, M. (2022, April). Privacy preservation and security management in VANET based to Software Defined Network. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 96-101). IEEE.

[74] Zhang, Y., Li, C., Chen, N., & Zhang, P. (2022, May). Intelligent requests orchestration for microservice management based on blockchain in software defined networking: A security guarantee. In *2022 IEEE International Conference on Communications Workshops (ICC Workshops)* (pp. 254-259). IEEE.

[75] Akilandeswari, V., Kumar, A., Thilagamani, S., Subedha, V., Kalpana, V., Kaur, K., & Asenso, E. (2022). Minimum latency-secure key transmission for cloud-based internet of vehicles using reinforcement learning. *Computational Intelligence and Neuroscience*, *2022*.

[76] Yang, G. (2022). An overview of current solutions for privacy in the Internet of Things. *Frontiers in Artificial Intelligence*, *5*, 812732.

[77] Liu, S., Yu, J., Chen, L., & Chai, B. (2022). Blockchain-assisted Comprehensive Key Management in CP-ABE for Cloud-stored Data. *IEEE Transactions on Network and Service Management*.

[78] Ataei Nezhad, M., Barati, H., & Barati, A. (2022). An authentication-based secure data aggregation method in Internet of Things. *Journal of Grid Computing*, *20*(3), 29.

[79] Siddiqui, S., Hameed, S., Shah, S. A., Ahmad, I., Aneiba, A., Draheim, D., & Dustdar, S. (2022). Towards Software-Defined Networking-based IoT Frameworks: A Systematic Literature Review, Taxonomy, Open Challenges and Prospects. *IEEE Access*.

[80] Bhardwaj, S., & Panda, S. N. (2022). Performance evaluation using RYU SDN controller in software-defined networking environment. *Wireless Personal Communications*, *122*(1), 701-723.

[81] Tivig, P. T., Brumaru, A., & Obreja, S. G. (2022, June). Creating Scalable Distributed Control Plane In SDN To Rule Out The Single Point Of Failure. In *2022 14th International Conference on Communications (COMM)* (pp. 1-6). IEEE.

[82] Albu-Salih, A. T. (2022). Performance evaluation of ryu controller in software defined networks. *Journal of al-qadisiyah for computer science and mathematics*, *14*(1), Page-1.

[83] Li, J., Cai, J., Khan, F., Rehman, A. U., Balasubramaniam, V., Sun, J., & Venu, P. (2020). A secured framework for sdn-based edge computing in IOT-enabled healthcare system. *IEEE Access*, *8*, 135479-135490.

[84] F. Keti and S. Askar, “Emulation of software defined networks using mininet in different simulation environments,” in 2015 6th International Conference on Intelligent Systems, Modelling and Simulation, 2015, pp. 205–210.

[85] Banerjee, U., Vashishtha, A., & Saxena, M. (2010). Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection. International Journal of computer applications, 6(7), 1-5.

الخلاصة

الشبكات المعرفة بالبرمجيات (SDN) هي واحدة من افضل التقنيات التي تتيح إدارة الشبكة عبر البرمجيات وذلك عن طريق فصل عمليات التوجيه (مستوى التحكم) عن مستوى البيانات وجعل عملية التحكم في الشبكة من خلال جزء مركزي واحد، وهو المتحكم . حيث يتكون مستوى التحكم من وحدة تحكم واحدة أو أكثر ويعمل بمثابة عقل الشبكة. أدى هذا الفصل إلى العديد من الفوائد، بما في ذلك تقليل التكلفة وقابلية البرمجة والمرونة.

اضاف هذا الفصل مشكلة الامان في العديد من أجزاء الشبكة، وخاصة في مستوى البيانات، لأن الرسائل غير مشفرة، وبالتالي يمكن لأي مهاجم التنصت على الرسائل الحساسة والوصول إليها ومن ثم إسقاط الشبكة بأكملها.

تقترح هذه الرسالة طريقة لتشفير البيانات في مستوى البيانات وإدارة مفاتيح التشفير ديناميكيًا في بيئة SDN عن طريق تضمينها داخل الشهادات وتوزيع تلك المفاتيح عبر طبقة SSL الى جميع الاجهزة الموثوق بهم في مستوى البيانات. تم الاعتماد على خوارزمية التشفير Rivest–Shamir–Adleman (RSA) لتحقيق سرية البيانات المنقولة عبر الشبكة.

ونتيجة لذلك، ستصبح البيانات مشفرة، وستكون الشبكة أكثر أمانًا بحيث لا يتمكن المهاجمون من التسبب في أي كشف لبيانات الشبكة. ومن ناحية أخرى، نتيجة لاستخدام إدارة التشفير الديناميكي، لم يتدهور أداء الشبكة. أفضل النتائج لمتوسط استقرار إنتاجية نظام SDN الآمن المقترح هو ١٢,٣٦٦ جيجابت في الثانية ومتوسط الوقت هو ٨,١١١ ثانية من تدفق الحزم.

جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة بابل
كلية تكنولوجيا المعلومات
قسم شبكات المعلومات

إدارة مفتاح تشفير مقترحة في بيئة SDN

رسالة مقدمة الى

مجلس كلية تكنولوجيا المعلومات - جامعة بابل كجزء من متطلبات

الحصول على درجة الماجستير في تكنولوجيا المعلومات / شبكات المعلومات

من قبل

هدى حيدر محمد الحمداني

بإشراف

أ.د. وسام سمير عبدعلي بهية

١٤٤٥ هـ ٢٠٢٣ م