An enhanced encryption algorithm for free public wireless networks.

An enhanced encryption algorithm for free public Wireless Networks

By

Christopher Khosa

(211249226)

Submitted in fulfilment of the requirements for the degree

Master of Computing: Information Technology

in the

Department of Information Technology

Faculty of Information and Communication Technology

at the

Tshwane University of Technology

Supervisor: Dr. DP du Plessis

Co-supervisors: Dr. TE Mathonsi & Dr. TM Tshilongamulenzhe

March 2023

Declaration

I, Christopher Khosa, declare that this dissertation entitled: *An enhanced encryption algorithm for free public Wireless Networks* is my work. It is being submitted in partial fulfilment of the requirements for the degree of Master of Computing in the Department of Information Technology at the Tshwane University of Technology, Pretoria. I further declare that all sources cited or quoted are indicated and acknowledged through a comprehensive list of references.

Signature	Date
…….	……

DEDICATIONS

I dedicate this work to all my family members who have supported and encouraged me to continue with my studies.

ACKNOWLEDGMENTS

I thank God for giving me strength, and for guiding me to complete this task. Thereafter, I would like to express my sincere gratitude and appreciation for the generosity of my supervisors who contributed to the success of this study, Dr. DP du Plessis, Dr. TE Mathonsi & Dr. TM Tshilongamulenzhe: for their continuous professional guidance in keeping me focused on the research topic, for their ongoing support, constructive feedback, and motivation throughout the study. I have learned many lessons from them throughout my research study. It has been a great honour to work with them. Finally, I would like to thank my whole family, and all my friends, and colleagues for their invaluable support and motivation during this entire period.

PUBLICATIONS:

Portions of this research study have been submitted and accepted for publication in the following journal and conference proceedings:

Journal:

1. Christopher Khosa, Deon P. Du Plessis, Topside E. Mathonsi and Tshimangadzo M. Tshilongamulenzhe, “Improved Encryption Algorithm for Public Wireless Network”. Accepted for publication at the Journal of Advances in Information Technology.

Conference proceedings:

1. Christopher Khosa, Topside E. Mathonsi, Deon P. Du Plessis and Tshimangadzo M Tshilongamulenzhe, “Improved Encryption Algorithm for Public Wireless Network”. In the proceedings of the 2022 International Conference on Computational Science and Computational Intelligence (CSCI'22), 14 – 16 December 2022, Las Vegas, USA, Page 1144 – 1150, ISBN-13: 979-8-3503-2028-2, DOI 10.1109/CSCI58124.2022. Available at: https://american-cse.org/csci2022-ieee/csci2022.zip or https://ieeexplore.ieee.org/xpl/conhome/1803739/all-proceedings

ABSTRACT

Wireless networks afford numerous benefits for productivity, due to the ease of access to information resources. A network can now be set up and changed more quickly, with less effort, and for less money. However, wireless technology also creates new threats; and alerts the existing risk profile for information security. In Wireless Fidelity (Wi-Fi), security mechanisms such as encryption algorithms play a vital role. A large amount of memory and power is consumed by those algorithms. This research study therefore proposed a computation efficient secure algorithm (CESA) that reduces the high consumption of power and memory to efficiently secure public Wi-Fi networks. The proposed CESA was based on a hash-based message authentication algorithm. A digital signature algorithm (DSA) was accomplished to produce and verify signatures using the secure hash algorithm (SHA). The network simulation 2 (NS-2) tool was used to evaluate the various settings of each algorithm, including key generation time, encryption time, and decryption time. Through the simulation, it was demonstrated that the proposed algorithm CESA outperformed both the EDH and AES algorithms in terms of key generation time, encryption time, and decryption time. To generate the key, the proposed CESA algorithm took up to 59 seconds, while the EDH and AES algorithms took almost 90 seconds. To encrypt the data, the proposed CESA algorithm took about 98 seconds, while EDH and AES algorithms took almost 167 seconds. To decrypt the data, the proposed CESA algorithm took about 80 seconds, while EDH and AES algorithms took almost 160 seconds. Thus, the EDH and AES make CESA more robust against attacks and very rapid in handling encryption and decryption processes.

Keywords: Wireless Networks, Wireless Fidelity, Encryption Algorithms, Computation Efficient Secure Algorithm**,** Hash-based message authentication algorithm, Digital Signature Algorithm.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

3DES: Triple Data Encryption Standard

AES: Advanced Encryption Standard

APs: Access Points

ARP: Address Resolution Protocol

ASCII: American Standard Code for Information Interchange

AWK: Aho, Weinberger, and Kernighan programming language

BD_ADDR: Bluetooth device address

CBR: Constant Bit Rate

CESA: Computation Efficient Secure Algorithm

CPU: Central Processing Unit

DES: Data Encryption Standard

DHCP: Dynamic Host Configuration Protocol

DNS: Domain Name System

DoS: Denial of Service

DSA: Digital Signature Algorithm

DSS: Digital Signature Standard

EAP: Extensible Authentication Protocol

ECC: Elliptic Curve Cryptography

ECDH: Elliptic Curve Diffie-Hellman

EDH: Enhanced Diffie-Hellman

FTP: File Transfer Protocol

HTTP: Hypertext Transfer Protocol

IEEE: Institute of Electrical and Electronics Engineers

IP: Internet Protocol

LANs: Local Area Networks

LTS: Long Term Support

MAC: Media Access Control

MATLAB: MATrix LABoratory

MD5: Message Digest

NAM: Network Animator

NIST: National Institute of Standards and Technology

NS-2: Network Simulator 2

NS-3: Network Simulator 3

OMNeT++: Objective Modular Network Testbed

QualNet: QUALity NETwork

OPNET: Optimized Network Engineering Tool

OS: Operating System

OTcl: Object Tool Command Language

PSK: Pre-Shared Key

QoS: Quality of Service

RAND: Random number

RSA - Rivest–Shamir–Adleman

SHA: Secure Hash Algorithm

SSP: Secure Simple Pairing

TCL: Tool Command Language

TCP/IP: Transmission Control Protocol/Internet Protocol

TCP: Transmission Control Protocol

TKIP: Temporary Key Integrity Protocol

TLS: Transport Layer Security

UDP: User Datagram Protocol

VoIP: Voice over Internet Protocol

WANs: Wide Area Networks

WEP: Wired Equivalent Privacy

WIMAX: Worldwide Interoperability for Microwave Access

Wi-Fi: Wireless Fidelity

WLANs: Wireless Local Area Networks

WPA: Wi-Fi Protected Access

WPA2: Wi-Fi Protected Access 2

XNOR: Exclusive NOR

Chapter 1: Introduction

1.1 Background Information to the Study

Public wireless fidelity (Wi-Fi) networks play a significant role for many organizations and users. This is because users can connect to the internet from multiple places such as airports, malls, universities, among many others. The broad range of wireless networks means that transmitted data must be secured in order to ensure data availability, integrity, and confidentiality (Bhanot, 2015).

Today, several cryptographic algorithms for Wi-Fi networks have been proposed, such as advanced encryption standard (AES), triple data encryption standard (3DES) and data encryption standard (DES). These algorithms use the public key exchange and they are suitable for private Wi-Fi networks (Kaur & Kaur, 2017). DES, 3DES, and AES are among the most popular encryption algorithms (Shahin et al., 2020). DES uses one key with 64 bits. Three 64-bit keys are used by 3DES; while various (128,192,256) bit keys are used by AES. Various (32-448) bit keys are used in Blowfish. DES is the first encryption standard to be used by the National Institute of Standards and Technology. DES has a key size of 64 bits and a block size of 64 bits. However, the drawbacks of DES are that it can be broken using a brute-force attack; and it consumes a great deal of memory, which has made it an unsafe block cipher (Su et al., 2019).

3DES is an enhanced version of DES. While DES uses a 64-bit block size and a 56-bit key, 3DES uses a 64-bit block size and a key size of 192 bits. The method of encryption used in 3DES is similar to the one used in DES, but it is applied 3 times consecutively to improve the level of encryption and extend the average time period during which the encryption remains secure.

However, several studies have shown that 3DES is slower in terms of efficiency than other block cipher techniques (Kaur et al., 2017).

AES is also a block cipher (known as Rijndael). It has 128, 192, or 256-bit variable key lengths. AES supports variable key lengths of 128, 192, or 256

bits. It operates on 128-bit data blocks and employs different numbers of encryption rounds (10, 12, or 14) depending on the length of the key used. AES appears to be rapid and durable, and thus can be implemented on various platforms, particularly on small devices (Arya & Soni, 2018). The disadvantage of this algorithm is the high consumption of power and memory.

Cryptographic algorithms are an essential cornerstone of information security in public Wi-Fi. However, the review of literature has shown that these algorithms often consume high power, memory, and central processing unit (CPU) time (Bhanot, 2015). Thus, they are not suitable for public Wi-Fi, as they increase the computational complexity and resources overheads.

This research study therefore developed a computation efficient secure algorithm (CESA) in order to secure data over a public Wi-Fi network. The Diffie-Hellman algorithm and the hashing algorithm created the CESA to guard against man-in-the-middle attacks. The thinking is that Diffie-Hellman calculation is defenseless against man-in-the-middle assaults. The proposed algorithm enhances the performance of the network, secures the transmitted data, and reduces the consumption of power and memory to secure zones over Wi-Fi networks. The proposed algorithm adopted the technique of asymmetric encryption. This is because two keys are used in the asymmetric encryption. The private key is used to decrypt, while the public key is used to encrypt. Because it uses two keys, the asymmetric cryptographic approach is far more reliable than the symmetric cryptographic approach (Blerina et al., 2017).

1.2 Motivation of the Study

Information security has become an important issue in data communication. Internet and network applications are growing very rapidly, therefore protecting such sensitive data has become the demand of the day. Encryption currently serves as a solution, and plays an important role in information security systems. Encryption uses algorithms to scramble data into unreadable text

which can only be decoded or decrypted by running the decryption algorithm with the associated key.

These algorithms consume a significant amount of computing resources such as CPU time, memory and battery power, together with consumption time. Encryption algorithms are available in various settings such as different key sizes, different block sizes, etc. It is necessary to develop an algorithm that will reduce the high consumption of power and memory to efficiently secure the public Wi-Fi network; therefore this is the fundamental motivation for this study to be conducted.

In addition, I was motivated by the desire to provide an industry-specific solution and thus make a significant contribution to the telecommunications industry. My career has been driven by continuous learning; I wanted to learn more about Wi-Fi networks by conducting this research. This investigation was further motivated by an interest in improving my research skills, and learning how to professionally and logically solve research problems.

1.3 Problem Statement

Based on the literature review, various encryption algorithms have been designed and implemented to secure public Wi-Fi networks. However, the shortcoming of these algorithms is that they consume excessive power and memory. This affects the quality of services (QoS) within Wi-Fi networks. Most of the nodes in these networks are mobile devices, and therefore are normally not connected to a power supply. In general, the source of energy for mobile devices comes from the battery. The majority of existing data-security approaches have a complex security mechanism that consumes much memory and power; this eventually affects the performance and renders the public Wi-Fi network vulnerable to several types of attack (Wan et al., 2017). Therefore, it is critically important to design an encryption algorithm that will address the

problem of high consumption of power and memory; thus efficiently securing public Wi-Fi networks.

1.4 Research Questions of the Study

The main research question of this research study reads as follows:

How does one develop an enhanced encryption algorithm less resource-intensive for a Wi-Fi network?

To answer the main research question, the following research sub-questions were developed:

1. What are the existing encryption algorithms used for free public wireless networks?
2. How can the proposed encryption algorithm be designed to reduce the high consumption of power and memory to efficiently secure data over free public wireless networks?
3. What is the effectiveness of the proposed algorithm compared with the existing algorithms?

1.5 Aim and Objectives of the Study

The main aim of this research study was to develop an enhanced security algorithm that will be less resource-intensive for free public wireless networks.

To achieve the main aim of the research study, the following objectives were developed:

1. Identify and analyse the existing encryption algorithms used for free public wireless networks.
2. Design an encryption algorithm that reduces the high consumption of power and memory, thus efficiently securing data over free public wireless networks.
3. Measure the performance of the proposed encryption algorithm against existing algorithms.

1.6 Significance and Benefits of the Research Study

The proposed CESA algorithm was designed so as always to ensure secure data and reliability to all authorized individuals. The developed encryption algorithm will also benefit all organizations, businesses, and individuals using free public wireless networks to access their information daily over the internet.

1.7 Research Methodology

To achieve the main aim and objectives of this research, this research study applied an experimental methodology. The methods given below were used.

1.7.1 Literature review

The review of the literature was conducted using the information from the conference papers, journals, books, and other internet sources. The researchers conducted a review of other researchers that were related to this study so as to:

- Find several shortcomings in current research
- Comprehend a subject that has been established but is fragmented
- Obtain new research topics.

1.7.2 Modelling

In this research study, mathematical equations were used when designing the proposed solution.

1.7.3 Implementation

In order to implement the proposed encryption algorithm, a code/programme was written. The research study developed the proposed algorithm by integrating various existing encryption algorithms.

1.7.4 Simulation

The NS-2 tool was used to test the performance of the proposed algorithm. In order to design the network topology and perform network configurations, this research study applied tool command language scripts and C++, respectively.

The AWK package was coded to measure the average of the performance metrics used in this research study.

1.8 Contribution of the Research Study

1. The proposed CESA algorithm was designed always to ensure secure data and reliability to all authorized individuals over free public wireless networks.
2. The proposed CESA algorithm allowed mobile devices on Wi-Fi to generate a shared hidden session key.
3. The design of the proposed CESA algorithm was achieved by integrating various asymmetrical cryptographic techniques; thus the proposed algorithm reduces the high consumption of power and memory.

1.10 Ethical Considerations

No ethical clearance was required for this research study because computer simulations were used to test the proposed solution.

1.11 Organization of the Dissertation

The dissertation is organized as follows:

Chapter 2 presents the literature review and overview of related work. The chapter presents an overview of wireless networks and the various encryption algorithms. In addition, Chapter 2 discusses wireless network technologies, standards, and protocols.

Chapter 3 provides the system design and architecture, based on the proposed encryption algorithm. Furthermore, we present in Chapter 4 the experimental setup and the technologies used such as simulation tools, to implement the enhanced encryption algorithm for the Wi-Fi network.

Chapter 5 discusses the tested results and analysis of the proposed encryption algorithms that were measuring performance using network simulation tools. The proposed enhanced encryption algorithm for the Wi-Fi network was

compared with all the other encryption algorithms used to securely protect data over the internet. Chapter 6 summarizes the research study and provides future work.

1.12 Summary

In this chapter, the research study presented a background of wireless networks and reasons for individuals to implement this network in various areas adopting free public wireless networks. The system architecture and the expected results of the research study were discussed. Furthermore, the chapter discussed the significance and benefits of the study, as well as the research methodology. In support of the proposed enhanced encryption algorithm, a set of research questions has been defined, which the dissertation answers by its technical objectives.

In the next chapter, the research study provides a literature review and related work on the research study.

Chapter 2: Literature Review

2.1 Introduction

Encryption is a critical tool for maintaining the security of sensitive information. By converting plaintext into an unreadable format (ciphertext), encryption helps ensure the confidentiality of the information being transmitted or stored. In addition to confidentiality, encryption can also provide other security benefits, such as digital signatures, authentication, and secret key management (Taha et al., 2019). Thus, the purpose of adopting encryption techniques is to ensure the information's confidentiality, integrity, and availability. In addition, this prevents information tampering, and duplicating.

Public Wi-Fi networks often use various cryptographic algorithms to secure data transmitted over the network. Some of the commonly used algorithms include 3DES, DES, and AES. These algorithms are based on the public key exchange and provide a high level of security for sensitive information transmitted over the network. The public key consumes high power and memory since a public exchange depends on a temporary and mathematical key created to encrypt the data sent over unsecured internet channels (Shahin et al., 2020).

The rest of this chapter is structured as follows: Section 2.2 presents an overview of the cryptographic encryption algorithms. Section 2.3 presents encryption methods. Section 2.4 provides the background of encryption algorithms. Section 2.5 presents wireless network technologies. Section 2.6 presents wireless security threats and risk mitigation. In Section 2.7, the related work is presented. Finally, Section 2.8 provides the summary of the chapter.

2.2 Overview of Cryptographic Encryption Algorithms

Cryptography is the practice and study of information hiding and verification. It is usually referred to as the study of secrets when data is exchanged over the internet or other media. It is the art of protecting information from unauthorized

access by transforming it into a non-readable format, known as ciphertext (Taha et al., 2019). Only those who possess a secret key can decipher the message into plaintext. Cryptography consists of protocols, algorithms, and strategies that are used to consistently prevent or deny unauthorized access to sensitive information (Taha et al., 2019). Encryption processes consume high power and memory, causing poor network performance due to the high computational process of encrypting data.

The original message from sender to receiver is the “plaintext” and the message that is sent through the channel is denoted “ciphertext”. Cryptography has two processes: encryption and decryption. The process of creating plaintext to ciphertext is known as encryption. Reversing the process, to create plaintext from ciphertext is called decryption. The encryption process consists of an algorithm and a secret key. This key is used to secure encrypted data. Depending upon the secret key used, the algorithm will produce a particular output. If the secret key is changed, the output of the algorithm is also changed. Cryptography is divided into two categories, namely, symmetric key algorithm, and asymmetric key algorithm. Symmetric key cryptography uses the same key for encryption and decryption; while in asymmetric key cryptography two different keys are used: one is for encryption and the other for decryption (Panse & Kapoor, 2017; Hayouni & Hamdi, 2020).

2.3. Types of Encryption

The focus area in this research study is asymmetric encryption; therefore this section will only discuss asymmetric encryption. The asymmetric encryption method tends to be 1000 times slower than symmetrical encryption. This means that it can take 1000 times more CPU to process asymmetric encryption or decryption than would symmetric encryption or decryption (Logunleko et al., 2020). There is high computational complexity in the asymmetric key algorithm; therefore the algorithm consumes more power and memory (Bisht &

Singh, 2015). Hence, the aim of this study was to develop asymmetric encryption with less computational complexity in order to reduce the power and memory consumption. Figure 2.1 demonstrates the classification of major encryption methods.

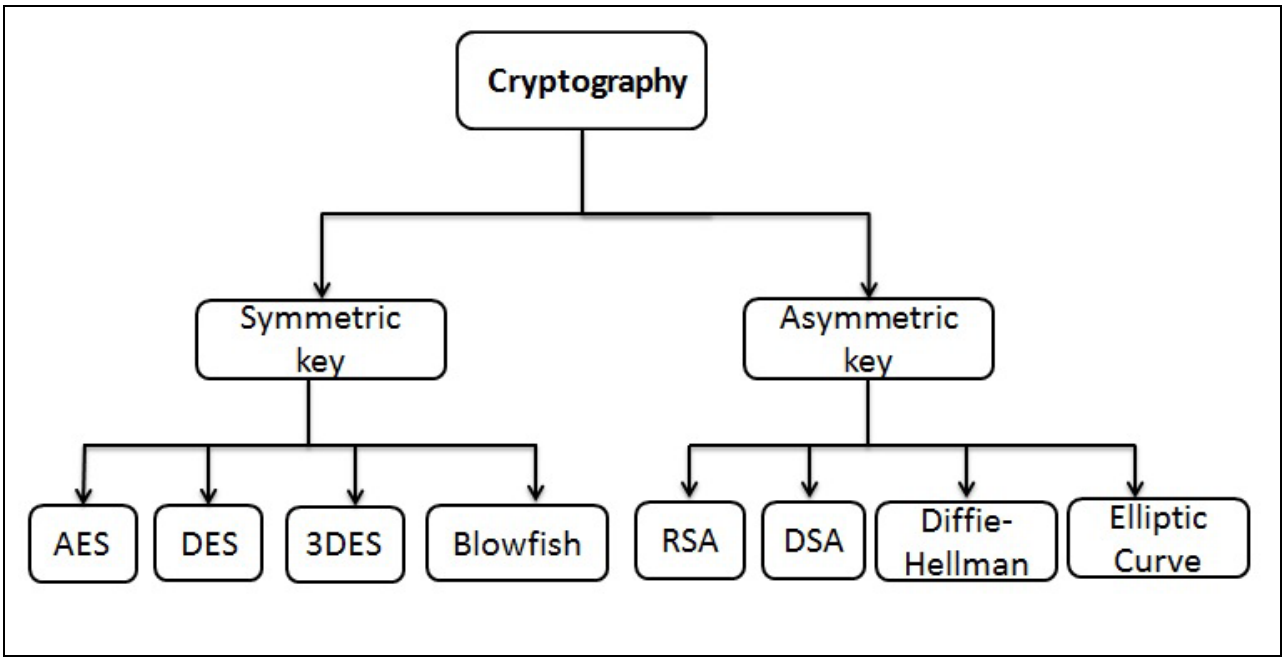

Figure 2.1: Classification of encryption algorithms (Bisht & Singh, 2015)

2.4 Asymmetric Encryption Algorithms

Of recent times, the internet has been providing essential communication between tens of millions of people. As the internet is increasingly being used as a tool for commerce, security becomes a tremendously important issue to deal with. More public Wi-Fi is thus deployed to allow multiple users to access the internet. However, public Wi-Fi is more vulnerable to attack than private Wi-Fi networks: this is because of the shortage of encryption on public Wi-Fi. Wireless data transmission is made possible by the manipulation of radio waves. By pulses of electricity, these waves are produced naturally. These radio waves can be changed to transmit sound or data through their amplitude or frequency (Yi et al., 2017). Public Wi-Fi networks have inherited the most common attacks from wireless networks, such as man-in-the-middle attacks. The

imposed restrictions on energy and power, as well as the exposure of the devices, make the public Wi-Fi networks more vulnerable to new threats, such as energy drain and Hello flood attacks.

There are many aspects to security and many applications, ranging from secure commerce and payments to private communications and protecting of passwords. One essential aspect of secure communications is cryptography, which is the focus area of this chapter. Cryptography is used to protect the information in public Wi-Fi networks. However, it is faced with many challenges such as higher power and memory consumption; hence it must be further improved. A brief review follows of various asymmetric encryption algorithms which make a great contribution to the field of cryptography:

A. Rivest-Shamir-Adleman (RSA)

The Rivest-Shamir-Adleman (RSA) algorithm uses both the public key and a private key. The RSA is a block cipher for digital signature algorithms or key exchange algorithms (Stallings et al., 2014). The RSA uses the variable-length key and the variable length encryption block. The message is encrypted by the sender; the receiver decrypts it. The message is encrypted with a public key and decrypted with the appropriate private key owned by the receiver. The RSA algorithm consists of three steps: generation of keys, encryption, and decryption. The RSA encryption technique cannot cope with the memory and power consumption for calculation, as it uses a tremendously large key (Yu & Kim, 2020). This has led to many potential attacks such as brute-force attacks, man-in-the-middle attacks, timing attacks, and selected ciphertext attacks (Stallings et al., 2014).

B. Diffie-Hellman Algorithm

The Diffie-Hellman is one of the first public key processes: it is a way of safely exchanging cryptographic keys (Stallings et al., 2014). In the Diffie-Hellman algorithm, the sender and receiver make a common secret key; then they begin

to communicate with one another through the public channel known to all. Diffie-Hellman secures a range of internet facilities. The sender must trust the public key cryptosystem when obtaining the recipient's public key and vice versa. This leads to significant consumption of computing resources such as CPU time, memory, and battery power. Therefore, it may also lead to the man-in-the-middle attack and DoS attack (Mandal et al., 2014).

C. Digital Signature Algorithm (DSA)

In 1991, the digital signature algorithm (DSA) was developed by the National Institute of Standards and Technology (NIST) United States, for use in its digital signature standard (DSS) (Nie et al., 2009). The DSA is based on the exertion of computing discrete algorithm difficulties (Stallings et al., 2014). DSA is used to authenticate and verify the integrity of digital signatures. DSA was accomplished to produce and verify signatures using the secure hash algorithm (SHA). If the sender desires to send a message to the receiver, the sender's signature generation uses its private key to generate a signature. Once the receiver receives a message, the signature verification uses the public key of the sender (Aufa & Affandi, 2018). The memory space, bandwidth, and power consumption are a constraint to the DSA process. Such can therefore lead to a session hijacking attack (El-Haii et al., 2016).

D. Elliptic Curve Cryptography

Elliptic curve cryptography (ECC) is based on the elliptic curve theory, and was developed by Koblitz and Miller in 1985. The ECC uses complex algebraic and geometric equations to generate a public key (Stallings et al., 2014). The ECC uses a private key to decrypt and generate signatures while using the public key to encrypt and verify signatures. The ECC consumes high energy and memory (Anerjee & Hasan, 2018).

Table I: Encryptions Parameters

Parameters	RSA	Diffie-Hellmann algorithm	DSA	ECC
Computation Time	Slow	Moderate	Moderate	Moderate
Memory Utilization	Requires more space	Requires moderate memory space	More memory space is required	Requires moderate memory space
Security Level	Poor security	Moderately secure	Moderately secure	Least secure

2.5. Wireless Network Technologies

In this section the research study presents the security protocols used in the wireless network technologies standards such as Bluetooth, Wi-Fi, Worldwide interoperability for microwave access (WiMAX); and long-term evolution (LTE) for protecting the authenticity, confidentiality, integrity, and availability of legitimate transmissions through the wireless propagation medium (Kui et al., 2016). For this study, the Wi-Fi and Bluetooth technologies will be discussed with a focus on Wi-Fi.

2.5.2. Wireless Network Technologies

A. Bluetooth

Bluetooth is a short-range and low-power wireless networking standard, which has been widely implemented in computing and communications devices as well as in peripherals, such as cellphones, keyboards, and audio headsets, among many others. However, Bluetooth devices are subject to many wireless security threats and may easily become compromised. Thus, for protection, Bluetooth has introduced diverse security features and protocols for guaranteeing its transmissions against potentially serious attacks.

For security reasons, according to Cao et al. (2013), each Bluetooth device has four entities, including the Bluetooth device address (BD_ADDR), private authentication key, private encryption key, and a random number (RAND), which are used for authentication, authorization, and encryption, respectively. More specifically, the BD_ADDR contains 48 bits, unique to each Bluetooth device. The 128-b private authentication key is used for authentication; and the private encryption key varying from 8 to 128 bits in length is used for encryption. In addition, RAND is a frequently changing 128-bits pseudorandom number generated by the Bluetooth device itself.

Bluetooth uses a secure simple pairing (SSP) encryption to authenticate devices and as encryption for keys that are shared. SSP is focused on cryptography of the public key, which conducts authentication between device communications through the visual confirmation of an integer value. SSP's primary objective is to ensure stable protection capable of preventing data-exchange intrusions. The ECDH public-key cryptography protocol is implemented by SSP to construct its link keys. Link keys are generated using private-public key pairs and the physical Bluetooth addresses of the communicating devices using the Elliptic-Curve Diffie-Hellman (ECDH) technique. This is because Bluetooth is designed for devices with limited resources. One of the challenges in Bluetooth networks is the higher power consumption caused by encryption mechanisms (Sun et al., 2017). This higher power consumption drains the batteries of the device; thus, it enables an injection-free attack to occur. The injection-free attack is based on compelling the key renegotiation of paired devices. The attack deploys a novel technique to force the key renegotiation of devices. Such a method exploits the properties of the bonding list of devices and their defences. Therefore the injection-free attack enables man-in-the-middle and DoS attacks to be carried out (Sun et al., 2017).

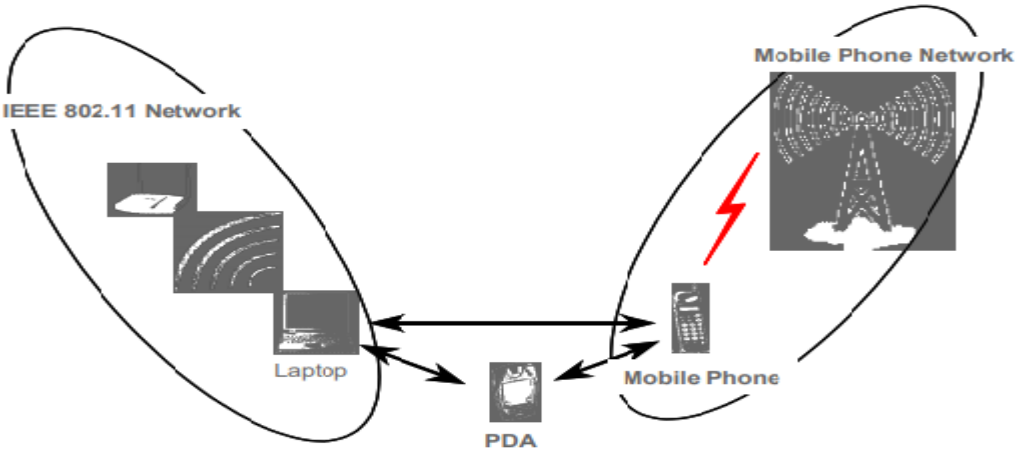

Figure 2 2: Bluetooth network

B. Wi-Fi

The family of Wi-Fi networks mainly based on the IEEE 802.11 b/g standards has been explosively expanding. The most common security protocols in Wi-Fi are referred to as wired equivalent privacy (WEP) and Wi-Fi protected access (WPA) (Danesh et al., 2010). WEP was proposed in 1999 as a security measure for Wi-Fi networks. Such privacy could make wireless data transmissions as secure as in traditional wired networks, as shown in Figure 2.2.

However, WEP is a relatively weak security protocol, and has numerous flaws. For instance, message integrity checking is ineffective, key management and updating is poorly provided, master keys are used directly, inter alia. Hence, the WEP can be “cracked” within a few minutes using a basic laptop computer (Yaacoub & Salman, 2020). As an alternative, the WPA was put forward in 2003 for replacing the WEP; while the improved Wi-Fi protected access 2 (WPA2) constitutes an upgraded version of the WPA standard (Badholia et al., 2018). Typically, WPA and WPA2 are more secure than WEP; and thus they are widely used in modern Wi-Fi networks.

The WPA was introduced to WEP insecurities as an intermediate solution. Only a subset of IEEE 802.11i has been introduced by WPA. WPA2 uses a particular AES mode known as the protocol counter mode cipher block-chaining message-

authentication code protocol (CCMP). CCMP offers both confidentiality of data encryption and integrity of data. A more reliable alternative to the RC4 stream cipher used by WEP and WPA is the AES. Two elements, the encryption and authentication of wireless LAN, are central to the WPA2 standard (Badholia et al., 2018).

The WPA2 encryption component mandates the use of AES, but for backward compatibility with the current WAP hardware, the TKIP (temporary key integrity protocol) is available. The WPA2 authentication component has two modes: enterprise and personal. The personal mode allows the PSK (Pre-shared key) to be used; and does not require users to be separately authenticated.

Enterprise mode, which requires individual user authentication based on the IEEE 802.1X authentication standard, uses the Extensible Authentication Protocol (EAP) to select from various EAP methods (Hidayat & Riadi, 2021). While there are more than five EAP methods available, the most common ones used in enterprise environments are EAP-Transport Layer Security (TLS), EAP-Tunneled Transport Layer Security (TTLS), Protected-EAP (PEAP), EAP-Flexible Authentication via Secure Tunneling (FAST), and EAP-Microsoft Challenge Handshake Authentication Protocol version 2 (MSCHAPv2) (Ghodke, 2016). According to Prasad & Manoharan (2017), EAP-TLS provides mutual authentication using digital certificates issued by a trusted Certificate Authority. EAP-TTLS encrypts the TLS tunnel between the client and server and supports other forms of authentication, such as passwords (Soliman & Azer, 2018). PEAP creates a secure tunnel for authentication without requiring client-side certificates, using a shared secret instead (Prakash & Kumar, 2018). EAP-FAST allows for streamlined authentication using previously established credentials, while EAP-MSCHAPv2 uses a challenge/response mechanism and is commonly used in Microsoft environments (Raghavendra, Ramesh & Chandrika, 2023). The selection of the EAP method will depend on the specific

needs of the network and the devices that will connect to it. In additional, the computation of these authentication algorithms consumes a substantial amount of computing resources such as memory and battery power (Panda, 2016). The consumption of power may affect the device. Switching it off while the session is running may lead to attacks such as session hijacking attacks, KRACK attacks, and man-in-the-middle attacks (Thankappan, Rifà-Pous & Garrigues, 2022). An attack is considered an active eavesdropping assault, mostly carried out by address resolution protocol (ARP) and cache poisoning on the media access control (MAC) layer (Shakdher, Agrawal & Yang, 2019). An attack aims to spoof the MAC addresses of the legitimate nodes; and claims to be the intended sender or receiver of traffic between two legitimate nodes by intercepting a link between two legitimate hosts on the network. Once the legitimate node is battery-exhausted due to battery consumption caused by higher computation, the attacker may attempt to take over the session (Shakhov & Koo, 2018).

2.6. Wireless Security Threats and Risk Mitigation

The high computational complexity of cryptography algorithms causes higher power and memory consumption. Such exposes the Wi-Fi network to threats such as man-in-the-middle, DoS, session hijacking, energy drain, Hello flood attacks, and other threats. This research study is focused on man-in-the-middle, DoS, and Session Hijacking in asymmetric algorithms mentioned in Section 2.4.

2.6.1 Denial of Services

Wireless Local Area Networks (WLANs) are characteristically defenceless against DoS: this attack may take place due to the expected death of a device caused by higher battery consumption. Everybody has the equivalent unlicenced frequencies, making rivalry inescapable in populated regions. As big business WLANs move to 802.11n, they can utilize diverts in the bigger, less-swarmed 5 GHz band, decreasing “incidental DoS”. Moreover, contemporary passages

access points (APs) can auto-alter channels to evade obstruction (Alblwi & Shujaee, 2017). Despite everything, DoS assaults occur in these forms: phony messages sent to disengage clients, expended APs assets, and keeping channels occupied. In order to mitigate common DoS attack methods like death floods, it is recommended to explore newer resources that address 802.11w management frame protection for enhanced security, as suggested by Alblwi & Shujaee (2017).

2.6.2 Man-in-the-middle attacks

Man-in-the-middle attacks between the network access point and network device are carried out by a hacker's third-party device. Once the battery of the legitimate network device dies due to battery consumption caused by the high computation of encryptions algorithms, the attacker can intercept the original route of data communication between the two legitimate network devices via this third-party device. Hackers can then acquire the authentication credentials of devices. Even worse, hackers might masquerade as or even change the information of legitimate devices, as an accepted and legitimate group (Shakhov & Koo, 2018).

2.6.3 Session hijacking attack

An attacker will suppress one of the machines, normally the client machines, and take over the role of the clients in the communication between the workstation and the server (Bharti & Chaudhary, 2012). The relationship between the two legitimate network devices is terminated by an ordinary intruder. The power consumption also suffers due to high encryption computation, which can cause the device to shut down unexpectedly. Such can lead an attacker to use several methods such as frame de-authentication using a spoofed media access control address to continue with the session.

2.7. Related Work

To offer more perspective on the performance of the encryption algorithms, this subsection describes and examines previous work conducted in the field of data encryption. The metrics taken into consideration are processing speed, throughput, power consumption, avalanche effect, packet size, and data types. This subsection also discusses the gaps identified in the previously proposed algorithms.

Pramono (2018) proposed a DES system for data protection. The DES cryptography method is a text data cryptographic method with a symmetrical type of key block cipher. The DES consists of two steps, namely, the encryption and decryption processes. The key used in DES cryptography has an effective length of 56 bits. The process of encryption using the DES cryptographic method consists of an 8-step process. The problem with this approach is that it is a complicated process and time-consuming because the length of the data being handled is longer.

Elminaamn et al. (2014) examined the performance of the symmetric encryption algorithms. The paper evaluates six of the most frequently used encryption algorithms, namely, AES (Rijndael), DES, 3DES, Rivest Cipher 2 (RC2), and Blowfish, as well as Rivest cipher 6 (RC6). A comparison of the data block sizes, various data types, battery usage, various key sizes, and ultimate encryption/decryption speed was performed at various settings for each algorithm. The following results were shown in the experimental simulation. When the results are seen either in hexadecimal base encoding or Basis 64 encoding, there is no important distinction. In terms of efficiency, modifying the packet size resulted in RC6 being less time-consuming than all other algorithms except Blowfish. However, when the data type was changed from text to image, RC2, RC6, and Blowfish were found to be less efficient compared to the other algorithms in terms of time consumption. In comparison, 3DES is still poor

compared with the DES algorithm. Finally, in the case of changing key size, it can be shown that a higher key size induces major battery and time consumption changes (only in AES algorithms or RC6).

Adekanmbi et al. (2015) assessed the performance of common encryption algorithms for throughput and energy consumption over a wireless networks. AES, DES, and BLOWFISH were the algorithms that were assessed by Adekanmbi et al. (2015). The quality and power consumption for wireless devices were evaluated using various text, audio, and image files. The results have shown that BLOWFISH performance was higher than that of AES and DES. However, BLOWFISH has a few drawbacks which include weak keys and plain-image insensitivity.

Mandal, Parakash, & Tiwari (2012) compared the two most widely used symmetrical encryption techniques, namely, DES and AES, based on the avalanche effect. This is due to one-bit variation in plaintext keeping the key constant; also to the avalanche effect due to one-bit variation in key keeping the plaintext constant; and the memory required for implementation and simulation time required for encryption. The avalanche effect is the property of any encryption algorithm in which a small change in either the key or the plaintext should produce a significant change in the ciphertext. The avalanche effect is very high for AES compared to DES; whereas memory requirement and simulation time for DES is greater than that of AES. This shows that AES is better than DES in terms of power consumption.

Pavithra & Ramadevi (2012) compared the performance evaluation of various cryptographic algorithms. Based on parameters taken as time, various cryptographic algorithms are evaluated on various video files. These video files have differing processing speeds on which the various sizes of files are processed. The time was calculated for encryption and decryption in the various video file formats such as Blob and .DAT, having file size from 1MB to

1100MB. Results showed that the AES algorithm is executed in a shorter processing time and on a better throughput level than on DES and BLOWFISH.

Arora et al. (2012) studied the performance of a range of security algorithms on a cloud network; and also on a single processor for differing input sizes. The researchers' aim was to find in quantitative terms the benefits of using cloud resources for implementing security algorithms; namely, RSA, message digest (MD5), and AES, which are used by many companies to encrypt large volumes of data. An assortment of video files with several processing speeds was used for the evaluation. The calculation of time for encryption and decryption in an alternative video file format such as .vob and .DAT was used, with file size from 1MB to 1100MB.

Results showed that the AES algorithm is executed in a shorter processing time and on a better throughput level than with DES and BLOWFISH. Increasing the block size enhances the level of security because it promotes greater diffusion. The block size represents the fundamental component of data that can be processed through encryption or decryption with one operation. A larger block size will lead to greater security, assuming all other factors remain constant. Generally, a block size of 64-bit has been considered a reasonable trade off and was nearly universal in block cipher design. The same block size of 64 bits was used for DES, 3DES, and Blowfish. The block size of AES is 128 and thus it is more secure. However, this method consumes more memory due to the larger block size.

Elhoseny (2016) developed the novel technique EGASON. For generating of a key, this technique uses an ECC; the genetic algorithm method is used for information encryption and decoding. Unlike symmetric key functions, this method suffers from brute force attacks and high computational costs when using ECC. This technique utilizes more power.

Shruthi (2016) proposed a key-management protocol based on the digital signature for secure data transfer. The key-management protocol proposed by

Shruthi (2016) was efficient in minimizing the rate of false data injection and the rate of a node failure; thus, the data to be transferred over the internet using a wireless network was secured. From the authentication of the users to the actual data being transferred, if any part of the data is intercepted, it was unreadable because of the strong encryption used. However, this method doubles the computational time which excessively consumes power for the device.

Tiwari & Kim (2018) proposed a new model for data encryption and decryption using ECC and deoxyribonucleic acid (DNA). DNA is cast off to allocate genes, which are used for the encryption of the data. The experiment is secured towards timing and power supply analysis (PSA) attacks. This technique consumes less power and memory; however, it is likely to be vulnerable to man-in-the-middle attacks.

Yue et al. (2019) proposed a hybrid approach. To encrypt plaintext blocks, this algorithm uses AES and ECC algorithms; then uses data compression technology to acquire blocks of ciphertext. After that, it links the ECC-encrypted MAC address and AES key to form a full ciphertext message. The hybrid technique reduced the encryption time and improved security; thus, it consumes less power. However, this technique is vulnerable to session hijacking attacks.

2.8. Summary

This chapter provided a detailed discussion on encryption algorithms and their types. The focus was directed on what encryption algorithms are; how they work, and their contribution to the lives of individuals in various areas that utilize the free public wireless network, to gain access to their sensitive information over the internet. The chapter provided the wireless network technologies that are currently being used by many businesses and organizations. However, there are new wireless network technologies on the

market that support today's network features. These networks can be used by businesses and individuals to help strengthen the security of the wireless network in public spaces.

The chapter also presented an overview of wireless network technologies that have been developed to support wireless network operation. The chapter presented various 802.11 standards. Chapter 1 introduced a set of research questions used to guide the research contained in this dissertation. The research questions deal with an encryption algorithm. It is therefore important to understand how to select the best encryption algorithm for information security protection. Furthermore, a brief overview was given of the research work previously conducted on encryption algorithms to protect data over the internet. The next chapter presents the contribution by this dissertation. The chapter illustrates the design of the proposed enhanced encryption algorithm to protect data using the free public wireless network. This is achieved by introducing the necessary network components and indicating the process flow of the proposed algorithm.

Chapter 3: System Architecture and Design

3.1 Introduction

This chapter presents the network architectural design and the implementation of the proposed CESA. This chapter begins by discussing the various system components and the building blocks in terms of the configuration of these components. Thereafter, this chapter achieves the goals of this study by providing discussions on the physical connections configured on the network. This includes the importance of each system component, the configurations, and the benefits that each component provides on the network to enhance the QoS in public Wi-Fi networks.

This study aimed at improving network efficiency by developing an enhanced security algorithm for public Wi-Fi networks. The proposed algorithm was developed to reduce the success of man-in-the-middle attacks in these networks. The proposed CESA design is mathematically presented in this chapter through advanced equations.

The rest of this chapter is structured as follows: Section 3.2 presents the system architecture; Section 3.3 presents the design of CESA; finally, Section 3.4 summarizes the chapter.

3.2 System Architecture

The proposed architecture has been constructed by integrating various network components such as APs and Layer 3 switch which connect to the server. In addition, the architecture includes the gateway configured to function as a bridge between the APs and the internet, as shown in Figure 3.1. The APs are configured to be wirelessly connected to the switch, which performs transmission and thus operates at the distribution layer of the network. The switches have applied MAC addresses to define the devices that are attached to them. The transmitter helps in the control and configuration; and promotes a centralized network. The transmitter facilitates the connection between APs and

the server, and helps with external communication. The Layer 3 switch facilitates communication between clients and service providers. It can perform routing and bridging tasks simultaneously. In addition to this, it has many Ethernet ports, and is easier to configure. It can connect many client machines, and normally costs less than other switches. Furthermore, it offers advanced QoS features including packet prioritization, classification, policing, tagging, queuing, and scheduling. In this work, therefore, the Layer 3 switch is employed to promote prioritization to guarantee a certain level of QoS. The server is configured with a static IP address which is used as a domain name system (DNS) address. The server represents the local cloud to host archives, databases, and security settings. Aps, on the other hand, are configured with unique static IP addresses. Meanwhile, the dynamic host configuration protocol (DHCP) automatically allocates IP addresses to the client machines.

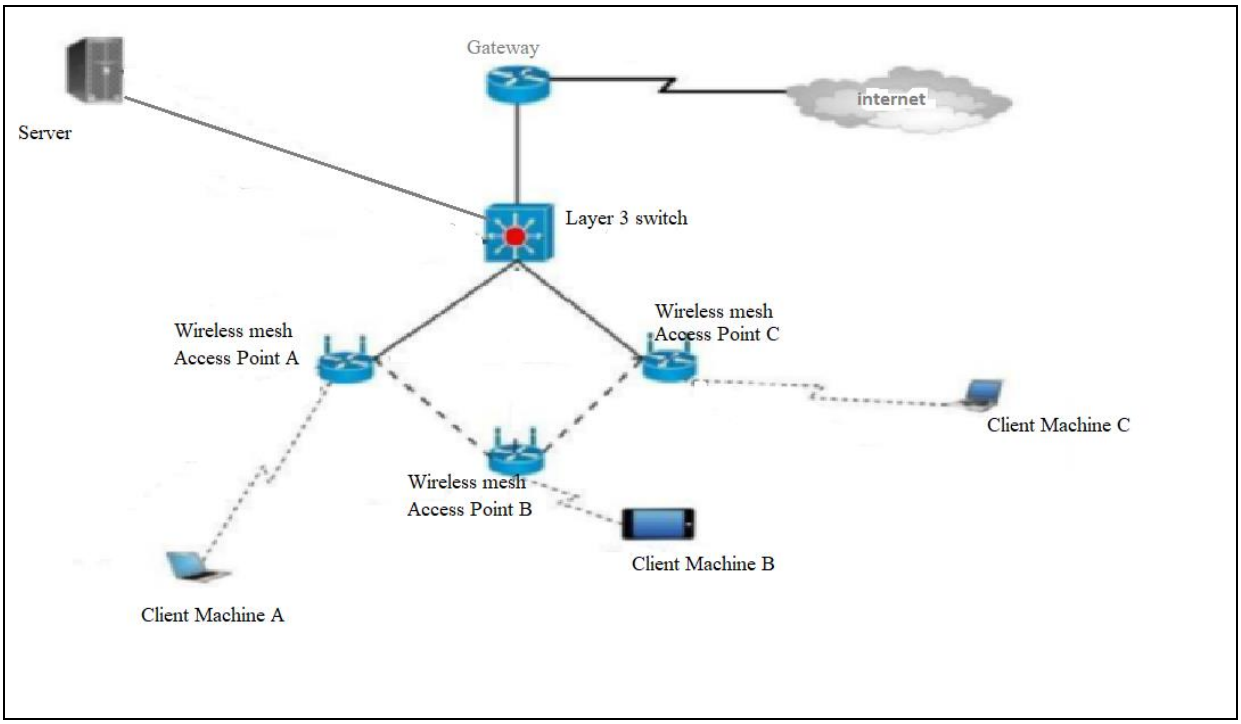

Figure 3.1 Typical wireless networks architecture

As shown in Figure 3.1, the proposed network architecture will be mathematically modelled as a directed graph of $Y = (M, L)$ with $M = \{M_1, \dots, M_i, M_{i+1} \dots, M_k\}$ calculating and defining the total number of client

machines. Meanwhile, $L = \{L_1, \ldots, L_i, L_{i+1} \ldots, L_K\}$ calculates and defines the number of connection links to aid in facilitating the communication between client machines and APs within the network and remote communications.

3.3 Computation Efficient Secure Algorithm

In this study, CESA was proposed to reduce memory and power consumption, and response time in public Wi-Fi networks. The proposed CESA was designed by the hashing algorithm and the Diffie–Hellman algorithm in order to prevent man-in-the-middle attacks. This is primarily because the Diffie–Hellman algorithm is vulnerable to man-in-the-middle attacks, and thus, hashing algorithm aids in reducing and preventing those attacks. The purpose of the hashing algorithm is to generate authentication encrypted keys. The algorithm ensures that the process of encryption and decryption of the keys generated is carefully considered to prevent man-in-the-middle attacks encountered by the Diffie-Hellman algorithm.

3.3.1 Diffie–Hellman Algorithm

As mentioned in Section 2.4, this algorithm is used to establish a shared secret to communicate secretly, meanwhile exchanging data over public networks using the elliptical curve to generate points for the secret key (Stallings et al., 2014). The Diffie–Hellman algorithm steps are shown in Figure 3.2; and are as follows:

(a) There are four variables − one prime Z and H and two private values e and f.

(b) The variables Z and H are key numbers publicly available to be used to generate a key and exchange it publicly. Therefore, devices (for example, machine A or B) use those private values (e and f) thus generating keys to be exchanged publicly. The recipient receives the key which then generates a secret key. This, therefore, means client machines A and B will have the same secret key to encrypt.

Table II: Client Communications (A and B)

Client Machine A	Client Machine B
Public Keys presented = $\boldsymbol{Z}$ and $\boldsymbol{H}$	Public Keys presented = Z and H
Private Key chosen = $\boldsymbol{e}$	Private Key chosen = f
Key created $\boldsymbol{X = H^e mod\ Z}$	Key created $Y = H^f mod\ Z$
X and Y keys are exchanged	
Key acknowledged = $\boldsymbol{X}$	key acknowledged = Y
Created Secret Key = $\boldsymbol{Q_e = W^e mod\ Z}$	Created Secret Key = $Q_f = C^f mod\ Z$

Furthermore, $Q_e = Q_f$ have the same key to encrypt.

Step 1: Client machines A and B acquire public numbers Z = 23, H = 9

Step 2: Client machine A has chosen private key e = 4; meanwhile B has chosen private key f = 3

Step 3: Client machines A and B calculate public values

Client machine A: X = (9^4 mod 23) = (6561 mod 23) = 6

Client machine B: Y = (9^3 mod 23) = (729 mod 23) = 16

Step 4: Client machines A and B exchange the public numbers

Step 5: Client machine A acknowledges public key X=16 and B acknowledges public key Y = 6

Step 6: Client machine A and access point A compute symmetric keys

Client machine A: $Q_e = X^e mod\ Z$= 65536 mod 23 = 9

Client machine B: $Q_f = Y^f mod\ Z$ = 216 mod 23 = 9

Step 7: The generated 9 is the key for the machines to share.

However, as mentioned previously, the Diffie–Hellman algorithm is dangerously exposed to man-in-the-middle attacks. Consequently, client machines normally encounter these issues as a result of a lack of authentication (Mandal et al., 2014). In this study, therefore, it has been carefully considered that the creation of a key for the encryption must be secured to improve the security of the proposed algorithm.

3.3.2 Integrating hashing with Diffie-Hellman Algorithm

In this section, the integration of a hashing algorithm with the Diffie-Hellman algorithm presented is conducted to reduce and prevent the success of man-in-the-middle attacks on public Wi-Fi networks.

Hashing is used as a safe communication technique between the communicating parties for the longer term. Yang et al. (2019) explained the concept of converting the data human-readable content into an unreadable format. This, therefore, means that only authorized communicating parties will be able to reverse the data into readable format through the generated and shared keys.

Hashing is employed to overcome the success of man-in-the-middle attacks using hacking functions. This ensures that the information is shared or communicated among the authorized communicating devices or client machines.

The Diffie-Hellman algorithm faces a significant security challenge due to the inclusion of unsecured values such as Z, Z3, Z2, Z1, and H. To mitigate this problem, the algorithm calculates the hash value before transmitting data packets over the network. This process begins by translating values such as Z1 into binary form. Subsequently, the translation continues, and the count of 1s is computed and stored in a temporary variable, such as Temp. The values of Z1 and Temp are obtained through modulus division and appended to the binary value of the original Z1. By taking these steps, the security of the Diffie-Hellman algorithm can be improved, and the transmission of data packets can be better protected against potential threats.

Below are the steps on how the hashing algorithm is integrated into the Diffie–Hellman algorithm to generate key values:

Step 1: Client machines A and B choose two mutual parameters such as Z and H.

Step 2: The hash function then sends values to Z and H as shown below:

(i) Z and H values are translated to binary

(ii) *Count the 1s in the translated binary values*

(iii) *Save the count in a variable*

(iv) *Yield the value of the mod of Z*

(v) *Count the zeros*

(vi) *Take the sum of the characters from the count value obtained in the previous stage*

(vii) *Enter the original Z binary, the sum of the special characters, and finally, include step (iv) value when sending*

(viii) *The same procedure applies to H.*

Step 3: Direct this value to the recipient machine

Step 4: Client machine B will use the same method from Steps 2 i-vii to determine the hash value of both Z and H

Step 5: Client machine A computes $1 = H^e mod\ Z$

Step 6: The computing using measures from Step 2 i to vii creates a hash of $Z1$

Step 7: Client machine B then computes $Z2 = H^f mod\ Z$

Step 8: Then the client machine B generates a hash of Z2 using steps 2 to 7

Step 9: Both client machines A and B then authorize these values to each other

Step 10: The hash key is determined once more after obtaining these values at the appropriate position

Step 11. Client machine A finally computes Key $e = Z2^e mod\ Z$

Step 12: Client machine B finally computes Key $f = Z2^f mod\ Z.$

ALGORITHM 1: HASHING KEY IN DIFFIE–HELLMAN ALGORITHM

1. Begin
2. $Z = 0$
3. $H = 0$
4. $e = key$
5. q = base_value

6. i= integer
7. $f = key$
8. Set key[]=J
9. **WHILE** VALUE_COUNT = *0*
10. $Z = Z^e mod\ f$ & $H = H^e mod\ f$ //calculating (Z) &(H)
11. **IF** VALUE_COUNT < Z & H **THEN**
12. $Z = Z_i q^i$ & $H = H_i q^i$ then // Z & H converted to binaries
13. ***ELSE*** $Z = \sum_{i=1}^{Z} \binom{i}{z} e^i$ & $H = \sum_{i=1}^{H} \binom{i}{H} e^i$
14. Z= $(Z\ mod\ i)$ & H = $(H\ mod\ i)$ **THEN**

Client machine A & access point A selects a private number which is not a common thing on the network *(e.g. Client machine A chooses e , and access point*

15. ***ELSE*** *A chooses* f
16. **END IF**
17. **END IF**
18. **LOOP**
19. *VALUE_COUNT= 0*
20. **IF** Created-key = *Client machine A* = $Z1 = H^e mod\ Z$ **THEN**
21. Client machine B = $Z2 = H^f mod\ Z$ *//creating the key in a manner that*
22. *Client Machine* $A = Z1 = H^e mod\ Z$
23. *Wireless mesh access point A* = $Z2 = H^f mod\ Z$
24. **ELSE** Machine A and B key-values converted to binary **THEN**
25. Compute the number of 1s in client machine A Key
26. Compute the number of 1s in wireless mesh access point A Key
27. Created-key = 3
28. **ELSE**
29. Machine A key = (Machine B Key mod variable 1=)
30. Machine B Key = (Machine B Key mod variable 1=)

31. **ELSE** Key $e = Machine\ B\ key\ e\ mod\ Z$

32. Key $f = Machine\ A\ Key\ f\ mod\ Z\ )$

33. Key $e = Key\ f$

34. **END IF**

35. **END IF**

3.3.3. Encrypt and decrypt the keys

This section introduces the process of encryption and decryption applied to Algorithm 1 to efficiently curb out or reduce the success of man-in-the-middle attacks in public Wi-Fi networks. The encryption and decryption processes are used to encrypt and decrypt the key generated. This does not necessitate any additional complex steps; it is the encryption and decryption of the keys that were generated.

3.3.3.1 Encryption and decryption procedure

The encryption and decryption methods are shown in Figure 3.2. The plaintext is converted to American standard code for information interchange (ASCII) decimal values before being converted to binary values. In addition, the key that was created in Algorithm 1 is translated to binary. Both plaintext and key are XNORed after the binary values are obtained. The XNORed yields the ciphertext complex which becomes indecipherable because the attacker will never be able to know the key. Once the XNORed has been completed, the obtained indecipherable data is moved to the left. This will result in a new piece of indecipherable text. The received text is then subjected to 1s complement. The last phase involves breaking the temporary ciphertext into two subsets, which then switch positions. Thereafter, the finally acquired ciphertext is translated back to ASCII values and sent to the destination.

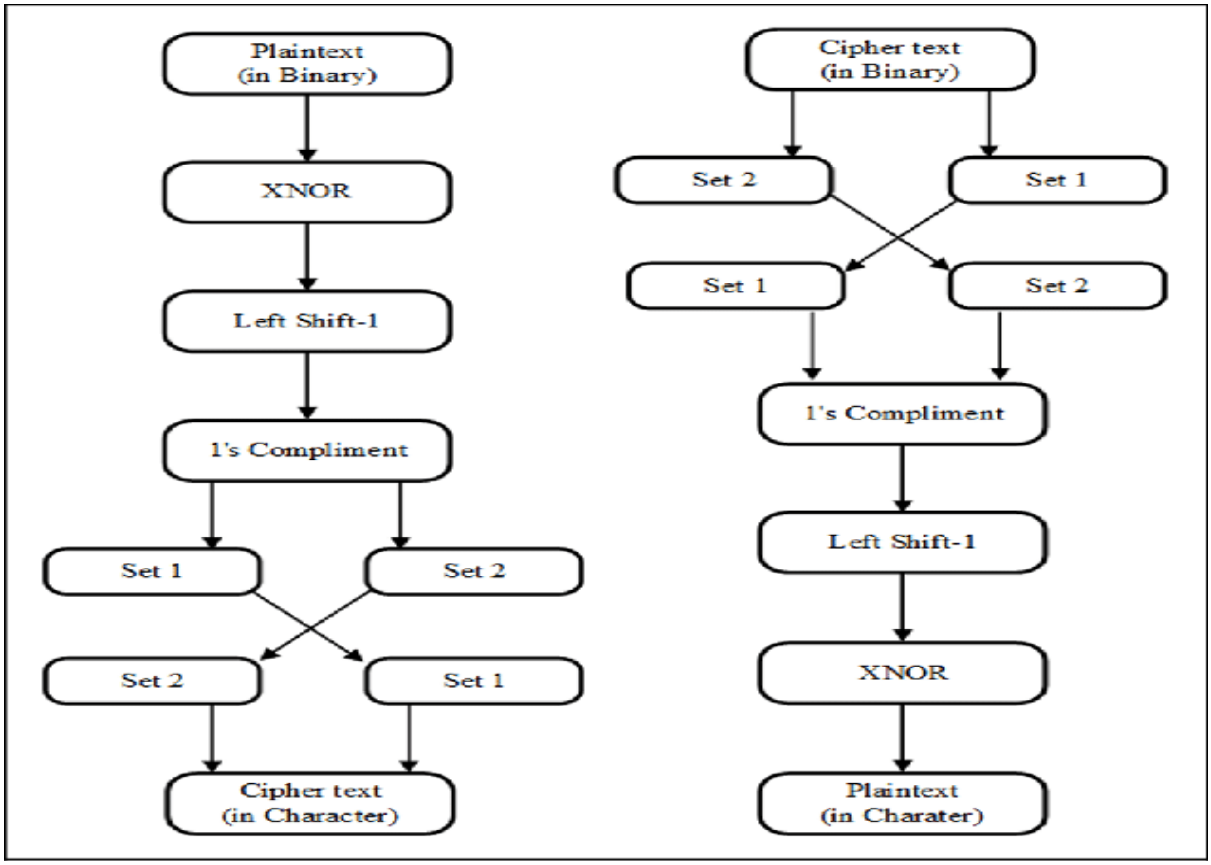

Figure 3 2: Decryption and encryption method (Mathur et al., 2018)

Since the plaintext is XNORed with the key, it cannot be broken unless the key is found. The key cannot be known unless one of the parties' hidden variables is known, which is impossible, since such information has never been transmitted over the network. Algorithm 2 shows how the encryption and decryption of keys are generated in Algorithm 1.

(a) Encryption phases

Phase 1: Client machine A decides the plaintext to forward to client machine B.

In Phase 2: every letter is converted to its corresponding ASCII decimal value. For instance, 'a' is represented as "97". Next, this decimal value is converted to an 8-bit binary value, such as "1100001" in the case of 'a'.

Phase 3: The binary value acquired in Phase 2 is then XNORed with a bit key length created in Algorithm 1.

Phase 4: Once the results from Phase 3 have been obtained, those results are then moved to the left-hand side.

Phase 5: The performance from Phase 4 is multiplied by the one's complement.

Phase 6: The performance from Phase 5 is split into two parts (e.g. 1100 = 11 and 00). These two parts then swap positions as the process progresses (e.g. 0011). This move alters the binary in general, making it more difficult to decipher the algorithm.

Phase 7: Once all the encryption phases have been completed, the final obtained value in Phase 6 is translated to ASCII decimal and then to an alphanumeric text.

(b) Decryption phases

The decryption method of the proposed algorithm is also covered in Algorithm 2. This phase is designed to convert the data's unreadable format to readable. The various stages of the decryption process are detailed as follows:

Phase 1: The client machine B receives ciphertext that has no meaning.

Phase 2: Each letter of the ciphertext is transformed into an ASCII decimal value after it has been received. Once the computing of the decimal value is completed, it is converted to an 8-bit binary value.

Phase 3: Phase 2 results are then divided into two parts (for example, 1100 = 11 and 00).

Phase 4: These two parts swap places (for example, 00 and 11 = 0011).

Phase 5: Requires the results obtained in Phase 4 to be represented as one's complement.

Phase 6: The results are then moved once to the right.

Phase 7: The client machine: A private key is XNORed with the result generated in Phase 6.

Phase 8: Once all the decryption phases have been completed, the final obtained value is translated to ASCII decimal and then to the character.

ALGORITHM 2: COMPUTATION EFFICIENT SECURED ALGORITHM

- **The process of encryption:**

INPUT: key, variable

1. Key generated from ALGORITHM 1.
2. Plaintext (already transformed into binary values)
3. m = dimension of the letter
4. **FOR** each letter 'y' in an assortment
5. **IF** y is the portion of assortment, **THEN**
6. Variable 1= XNORed (plaintext, key)
7. Variable 2= left swing respectively bit once in (Variable 1)
8. Variable 3=yield 1s' complement of (Variable 2)
9. Variable 4=split data into two parts in (Variable 3) & saves as:

   Set 0 = $\frac{m}{2}$ and set 1= $\frac{m}{2}$ to m
10. Ciphertext= exchange (Set0, Set1)
11. **ELSE**
12. Create a key first, then go to Phase 1
13. **END IF**

- **The process of decryption:**

INPUT:

1. Key # Hashing key in Diffie–Hellman algorithm
2. Ciphertext $ binary values$
3. m = dimension of the letter

OUTPUT:

1. Variable 4=ciphertext
2. **IF** key == existing **THEN**

Variable 3= split data in two parts in (Variable 4)

Set 0= $\frac{m}{2}$ and set 1= $\frac{2}{1}$ to m

3. Variable 2 = exchange (Set0, Set1)
4. Variable 1=yield 1s' complement of (Variable 2)
5. Variable = left shift each bit once in (temp1)
6. Plaintext= XNORed (plaintext, key)
7. **ELSE**
8. Create key first, then go to Phase 1
9. **END IF**

3.4 SUMMARY

In this chapter, the research study presented the design of CESA. The algorithm was designed to improve the Diffie–Hellman algorithm, it being vulnerable to man-in-the-middle attacks. The proposed CESA prevents the man-in-the-middle attack in the Diffie–Hellman algorithm by generating a key between the communicating devices. Furthermore, the proposed CESA includes encrypting and decrypting the key that was generated. The proposed CESA reduced the computational time; thus it consumes less battery, memory, and CPU processing power.

The implementation of the CESA introduced in this chapter will be presented in the next chapter. The NS-2 simulation tool will be presented in more depth. In addition, in the next chapter other similar simulation tools will be discussed, highlighting their benefits and disadvantages.

Chapter 4: System Implementation

4.1. Introduction

Network simulators are useful as they can normally evaluate the performance of various solutions before implementing them in real network infrastructure. This research study used the NS-2 to evaluate the performance of the proposed algorithm designed in Chapter 3. The NS-2 is classified as an object-oriented, discrete event simulator; it is also open-source. Thus, it is freely available on the internet.

The rest of the chapter is organized as follows; Section 4 .2 provides an overview of network simulation software. Section 4.3 discusses network simulator categories. Section 4.4 provides an overview of the NS-2. Section 4.5 discusses the evaluation of NS-2. Section 4.6 describes the encryption algorithms used in public Wi-Fi, while Section 4.7 focuses on CESA simulation. The chapter summary is provided in Section 4.8.

4.2 An Overview of Network Simulation Software

A genuine testbed experiment is expensive and time-consuming. As a result, it is critical to use system modelling to understand the protocol's behaviour and to evaluate the new idea before it can be implemented in the testbed; and, finally, in a real test (Leon et al., 2018). Furthermore, most encryption algorithms that are proposed for public Wi-Fi are designed to support hundreds of nodes. This makes it difficult to implement and test their performance on the testbed without first confirming their feasibility and outcomes through modelling practices.

A simplistic demonstration of an actual system is used in system modelling. In this manner, the likelihood of a system's behaviour can be seen without implementing it in the real physical resources (Quarton et al., 2018). To study the behaviour of the system, various parameters can be used, often with some simplification assumptions. Modelling can be classified into two approaches: analytical, and simulation methods (Quarton et al., 2018).

The analytical modelling approach is used to mathematically characterize the system and incorporate computational solutions to provide a specific view of the system. The circumstances, parameters, and conclusions are acceptable, the study findings being based primarily on mathematical evidence. Analytical modelling is much less costly, more effective, and gives a broad picture of the effects and correlations of different indicators.

An analytical modelling methodology, such as stochastic Petri nets and process algebra, was used to analyse the performance of communication systems. The number of public Wi-Fi analytical studies, on the other hand, is restricted (Iskounen et al., 2016). Since cryptography is difficult to integrate into these analyses, this computational modelling limitation on public Wi-Fi is realized. This is because most of these analytical experiments indicate that the devices are not stable. In public Wi-Fi, security is one of the most important features of the network. This makes it difficult to use analytical modelling to investigate the network output.

It was problematic to use mathematical formulations, the scheme being vast and complex. Simulation is used to apply the solution in these situations. Since almost all the specifics of the system requirements can be unified into a simulation model, simulation usually needs less simplification than analytical modelling. Simulation outcomes, on the other hand, are not always thought to be as reliable as empirical results. Simulators are commonly acknowledged as a useful method for researching complex networks. Simulators have proved to be effective tools for evaluating device performance. The proposed model can be observed under different scenarios and conditions before it is designed using a simulator (Iskounen et al., 2016).

4.3. Classifications of Network Simulators Software

There are many tools available for network simulation, whether commercial network simulators such as QualNet, MATLAB, and OPNET, or open-source

network simulators such as NS-2, Network Simulator version 3 (NS-3), and OMNeT++. Each network simulation has strengths and weaknesses. The best choice is dependent on the platforms, types of tools, and user interfaces (Campanile et al., 2020). The network simulation tools are examples of time-dependent simulation.

These simulation tools maintain a simulation clock to monitor simulation time as it progresses through simulation events. A time-dependent simulation can be divided into two types: simulations driven by time, and simulations driven by activities. Actions are performed at all-time intervals of time units in a time-driven simulation. An activity simulation, on the other hand, does not execute activities at a set time period and at any illogical time. The activity with the shortest timestamp is collected, processed, and transferred to the next occasion's timestamp (Campanile et al., 2020).

4.3.1. Simulation tool selection methods

The NS-2 is the most widely used in the research community because it is open-source and freely available on the internet (Ramamoorthy et al., 2021). This is an example of an event-driven simulation tool. NS-2 provides network and protocol emulation for both wired and wireless networks. Because of its versatility and modulation capabilities, NS-2 has become extremely common among researchers, globally (Kumar et al., 2020). In this research study, the NS-2 tool was chosen to investigate the efficiency of cryptography algorithms in public Wi-Fi.

4.3.1.1. Objective Modular Network Testbed (OMNeT++)

OMNeT++ is an open-source simulation instrument of discrete C++ language (Varga, 2019). OMNeT+ + is not only used for network simulations, but is rather modular object-oriented in comparison with other simulators. It can also be used to assess and model the complicated performance of computers. Modules can have parameters to be used based on three main needs:

customizing the behaviour of the module, creating flexible topology models (where parameters, and link structures may be defined), and sharing variables for the communication modules. The simulator can be used as a model. According to the OMNeT++ user guide, a simulator can be used for modelling purposes such as computing networks and traffic modelling, multiprocessing construction, and distributed schemes (Varga, 2019).

OMNeT++ also provides smart and extensive GUI assistance. It can run on both UNIX and Microsoft Windows operating systems. Figure 4.1 below shows the OMNeT++ simulator interface. However, additional effort is needed to create OMNeT++ simulations in comparison with other simulators and they are therefore not used for this research study.

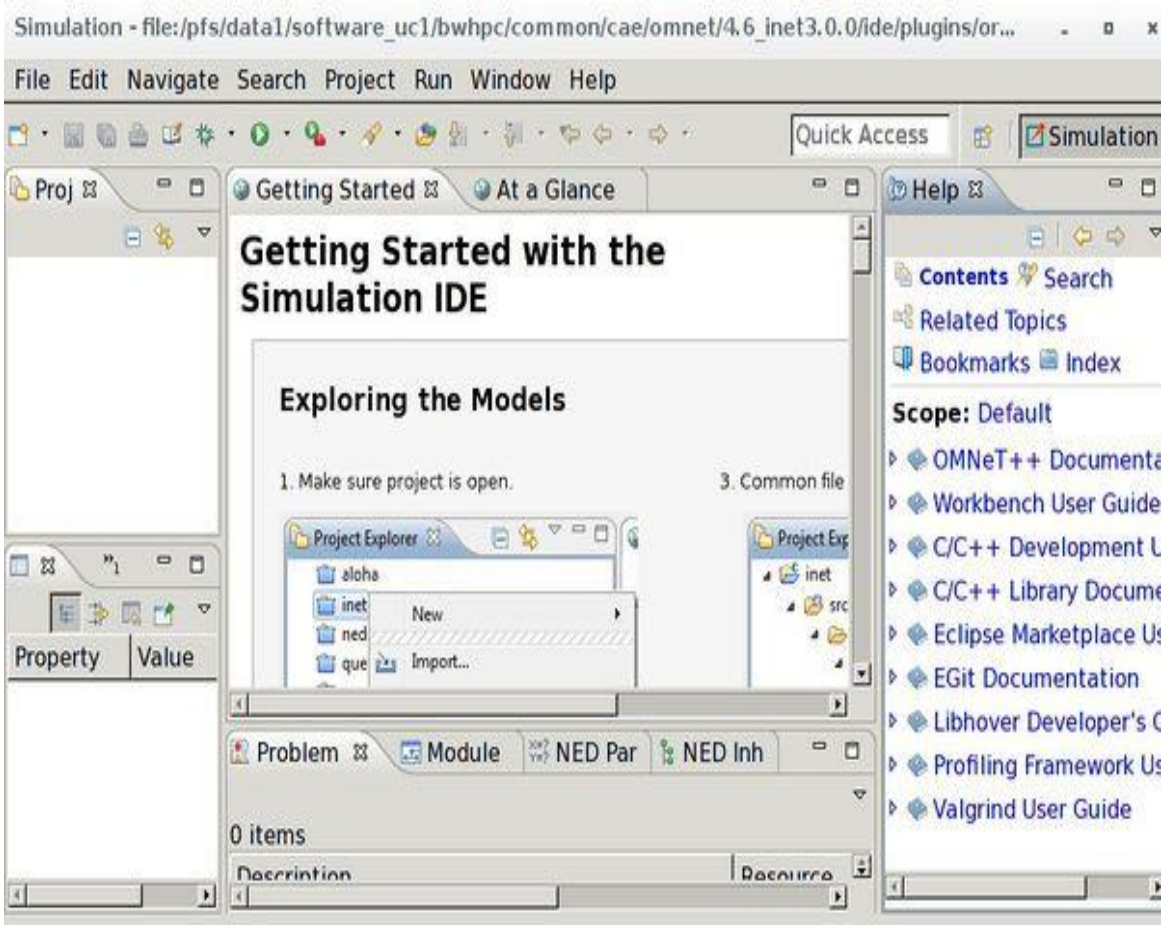

Figure 4.1: OMNeT++ simulator interface

4.3.1.2 Qualnet network simulator

A Qualnet network simulator is a discrete event simulator for large, heterogeneous networks and distributed apps running on them. With all of its parts for custom networking and simulation projects, the Qualnet's unparalleled speed, scalability, and loyalty make the optimization of current networks through quick modelling and in-depth analysis instruments easier for scientists (Huibo et al., 2019).

Wired LANs, WANs, mobile ad-hoc, wireless, satellite, WLANs, VoIP, telnet, FTP, and HTTP networks are all available in the Qualnet simulator. Rather than being open-source code, the Qualnet simulator is a business simulator. However, Qualnet is difficult to set up and use when combined with other resources; therefore it will not be utilised in this research study. The Qualnet simulator interface is illustrated in Figure 4.2.

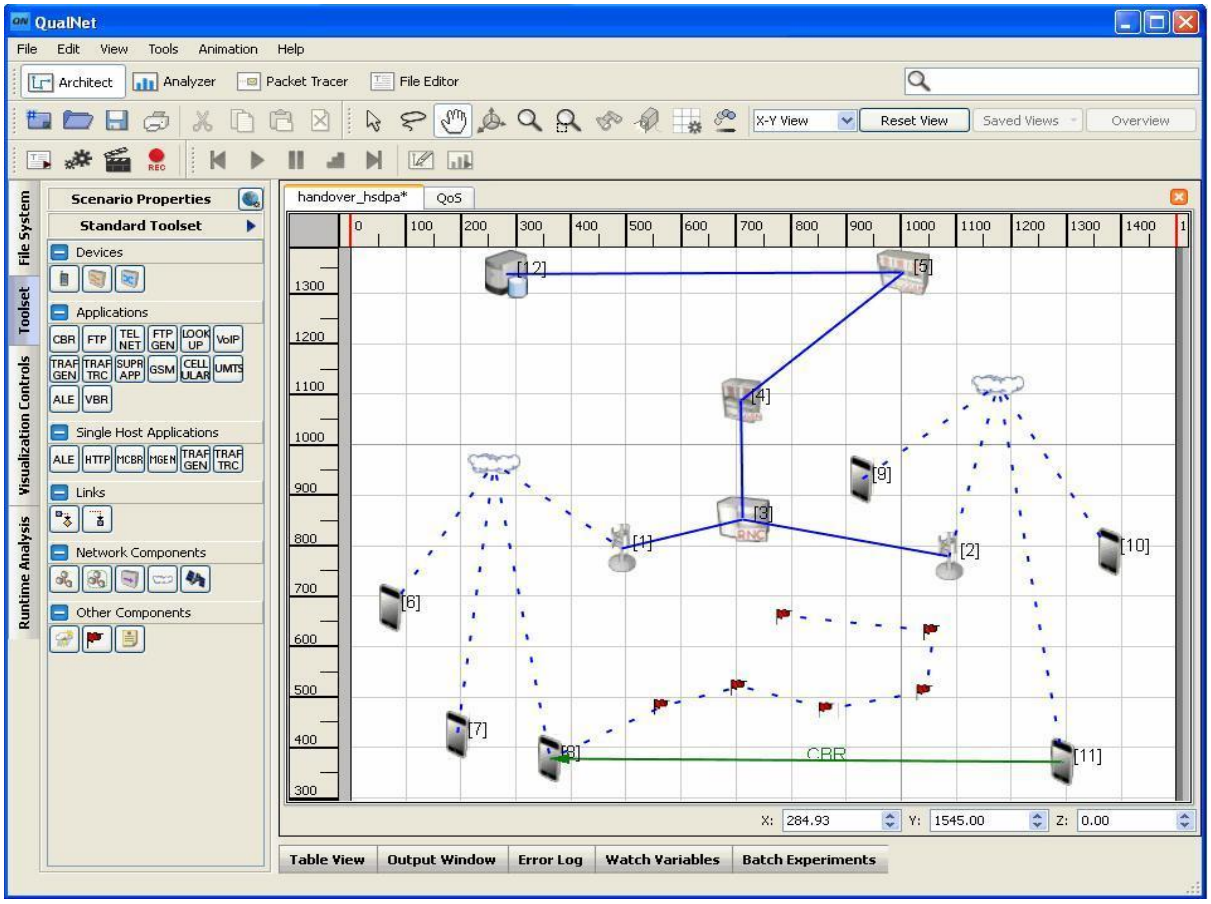

Figure 4.2: Qualnet simulator interface

4.3.1.3. Network Simulation 3 version

NS-3 is written in the C++ programming language. Unlike in NS-2, modern NS-3 anthology time hardware capabilities have not caused a challenge. The NS-3 does not rely on tool command language (OTcl) scripts to control the simulation; it can now be built entirely in C++. A modelling script, which is not possible in NS-2; it can also be composed as a C++ programme. NS-3 gains from Python's constrained scripting and visualization support. The C++ code is still accessible on NS-3 in conjunction with memory-management features such as delete, new, and malloc (Petersen et al., 2019).

A packet in NS-3 consists of a single byte buffer corresponding to the stream of bits sent over the real network. Furthermore, the packet contains data, which is added using subclasses; a header that adds input at the beginning of the buffer, and a trailer that adds data at the end. For simultaneous performance, NS-3 employs PyViz, a Python-based real-time visualization package. NS-3 was created to improve on NS-2. NS-3 does not support NS-2 written simulation projects. Figure 4.3 below shows the NS-3 interface.

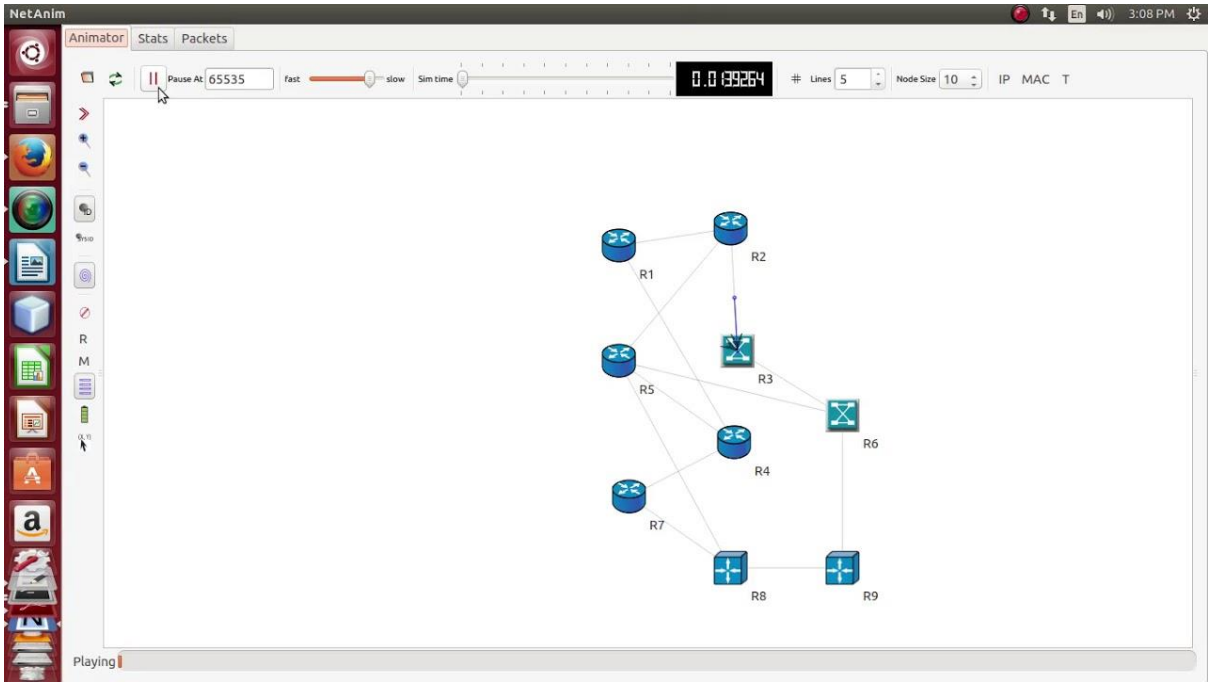

Figure 4.3: NS-3 interface

4.3.1.4. MATrix Laboratory (MATLAB) simulation

MATLAB is defined as a software package designed specifically for scientific calculations in mathematical software programming. MATLAB is widely used in a variety of fields, including applied mathematics, education, college, and industrial research (Ozgur et al., 2017). MATLAB can be used to alleviate algebraic expressions and equations of variations. The simulator can also be used to integrate numbers such as computational geometry. In addition, MATLAB has a powerful graphic tool that enables greater images to be generated in 2D and 3D. MATLAB has some tool kits in the fields of study for signal handling, stats, partial differential equation solutions, photo processing, analysing, and optimization (Tian et al., 2017).

MATLAB is a computer language of high performance that uses C++, C, Java, Python and Fortran. It has a MuPAD engine that can access the functions of the computer. The main problem with MATLAB is that basic MATLAB directives such as Linux, Microsoft Windows, and Mac systems must be properly understood. Furthermore, sophisticated indulgence is also needed for features such as 2D and 3D graphs, algebraic and differential equations, matrix calculations, and linear equations. MATLAB is not used in this research study, MATLAB having been designed mainly to solve complex numerical challenges (Kharab & Guenther, 2018). Figure 4.4 below illustrates the MATLAB interface.

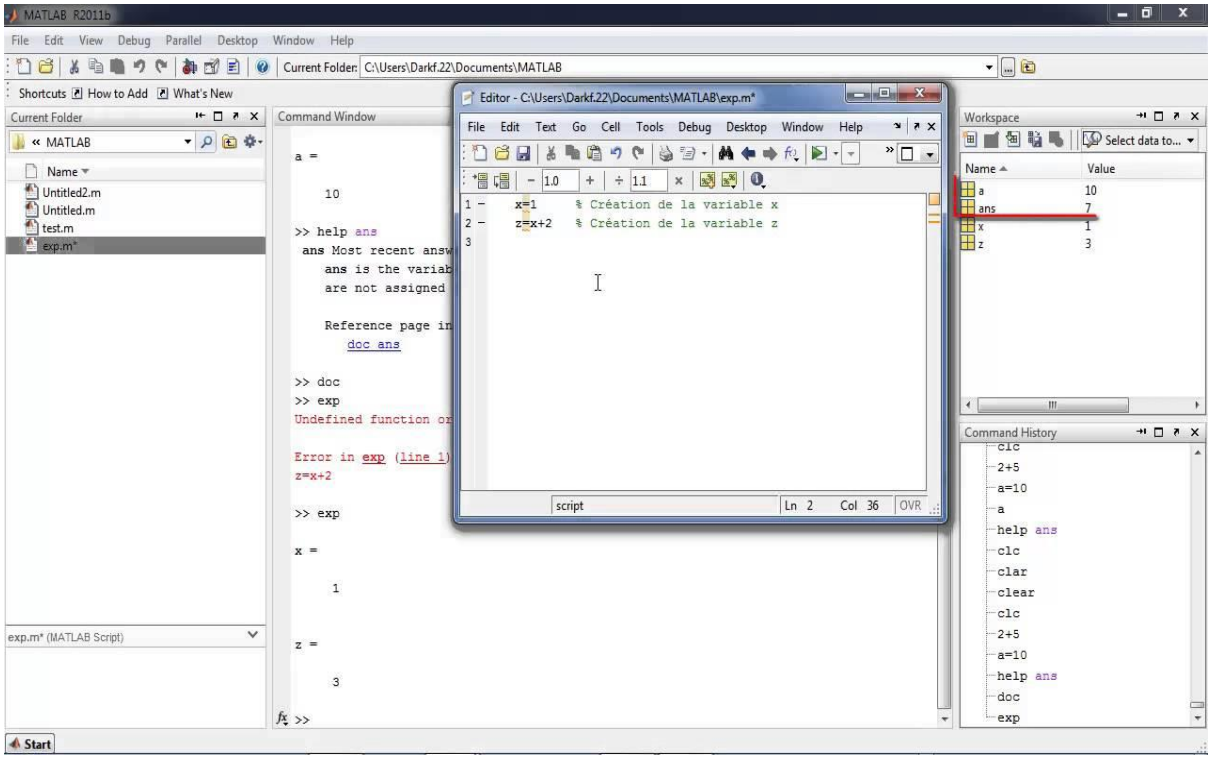

Figure 4.4: MATLAB interface

4.3.1.5. Network Simulator 2 (NS-2) tool

The NS-2 simulator uses C++ software to model the operation of nodes and scripts. In university studies, NS-2 is commonly used. This is because NS-2 allows several charitable groups to help improve the situation for the future. The simulation interface uses the OTcl and C++ for network topology and algorithm implementation, respectively (Al-Shidi et al., 2018).

The added value of NS-2 is the network animator (NAM) which has the interface to play, pause, speed, and avoid simulation during the evaluation stage. NS-2 is a network simulator that was developed for object-oriented discrete events at the University of Colombia-Berkeley. This means that at specified times NS-2 starts sending packets, which end at certain times. The NS-2 system implements various protocols such as TCP and UDP; traffic sources such as CBR Telnet, and FTP; and queue control mechanisms, for instance, Drop-Tail and algorithms for routing (Al-Shidi et al., 2018).

NS-2 is the most widely used open-source network simulator. It can investigate the behaviour of both existing and new protocols when simulating network

services and protocols for both wired and wireless systems. NS-2 makes use of the NAM package, which is a Tcl-based animation scheme for creating a visual representation of a network for production purposes. In Figure 4.5 the research study illustrates a standard graphic of a NS-2 simulator.

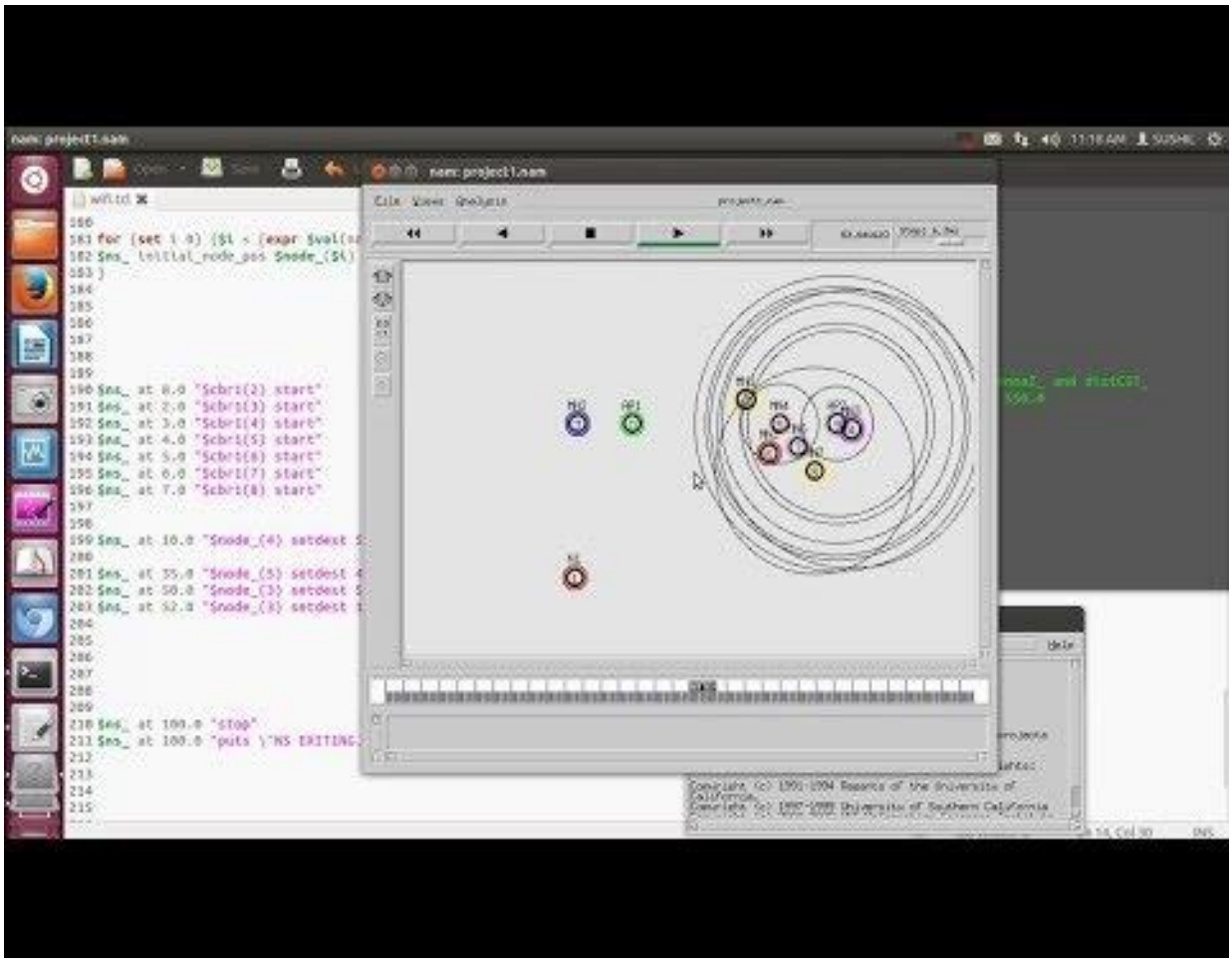

Figure 4.5: Standard graphic for the NS-2 simulator

4.4. Evaluation of Network Simulator

In the following Table 4.1, the network simulators are compared and summarized.

Table III: Contrasting Structures of Network Simulators

FEATURES	MATLAB	OMNET++	QualNet	NS2	NS3
Ease of Set-up	Moderate	Moderate	Hard	Easy	Easy
Graphical User Interface (GUI)	Yes	Yes	Yes	Yes	Yes
Programming Language	C++, C, Java, Python, & Fortain	C++	C	C++	C++
Accessibility	Yes	Yes	Yes	Yes	Yes
Scalability	High	High	High	Poor	Poor
Ease of Use	Moderate	Hard	Hard	Moderate	Hard
Larger Networks	yes	yes	yes	No	No

The NS-2 simulator tool is used to evaluate the CESA in this research study. The internet community offers novice researchers advice on the use of the NS-2 simulator. In Chapter 2, the NS-2 tool was used to analyse the alternatives suggested by a comparison of research studies. NS-2 supports all IEEE 802.15.4 simulations, as well as other IEEE 802.11 models. NS-2 also supports several wired and wireless network routing protocols. NS-2.35 was used in this research study to carry out all simulations. NS-2 provides a wide range of encryption, a variety of accessible designs such as mobility model, energy model, motion and traffic patterns, and wireless network interfaces. It also involves creating custom applications and protocols and changing multiple parameters for different layers. Thus, the NS-2 was used in this research study to simulate the performance of the CESA.

4.5. Network Simulator 2 Environment

The NS-2 is a discreet event network simulator that is widely used in the networking community. It was originally developed as part of the virtual internet testbed at California University, Berkeley (Silmi et al., 2020). NS-2 is

an object-powered network simulator. The programming languages used are C++ and OTcl, which are Tcl-script built on MIT objects. To improve performance, the control path is used to separate the data path execution. To reduce package and event processing time, the event scheduler and fundamental network component objects on the data path are written and compiled in C++. The missing feature in the C++ puzzle is OTcl. This shows that combining these two languages is extremely beneficial. C++ uses a robust protocol, while customers use OTcl to track and schedule the simulation situation.

The OTcl script activates the event planner, establishes network topology, and informs the traffic source how the event planner can start and stop the packets. The scenes can be changed simply by programming in the OTcl script. When a user is interested in creating a new network object, the new object may be written or assembled from the existing library by the user, which will plumb the data path.

One of the NS-2 main features is the ability to produce several random scenarios for movement and traffic patterns. NS-2 contains all major features, including abstraction, viewing, emulation, and traffic generation and scenarios. NS-2 is a comprehensive supporter of encryption algorithms such as AES, DES, Diffie-Hellman, Triple DES, RSA, among many others. The Diffie-Hellman algorithm is used in this research work.

In addition, C++ is used for reverse ends, and OTcl is used as a front end for the actual simulation (user interface). The reason is that C++ is suitable for the development and implementation of a network; however, the representation in these two languages is not visual or graphical. OTcl is graphic and descriptive, on the other hand, which makes the various parameters easy to review and alter.

In addition, since OTcl is an interpreted language, changes to the code do not need to be compiled. C++, on the other hand, necessitates compilation for connecting and creating an executable file following source code modification. As a result, OTcl changes are much quicker, albeit time-consuming to execute

because interpretations must be conducted over time. As a result, NS-2 benefits from C++ and OTcl, having the advantages of both languages.

For efficiency, NS-2 implements the data control path and the traffic data path separately. C++ is used to compile the event scheduler and the basic network objects. The OTcl script is used to start an event scheduler that controls the source node to start, finish, and set up the network topology in the source node (Nampally et al., 2017).

4.5.1. Network Simulator 2 architecture

As stated in Section 4.3.1.5, NS-2 is a C++ object-oriented simulator with a front end OTcl translator. The NS-2 supports a hierarchy of C++ classes and a separate hierarchy of OTcl interpreters. At packet-level, a packet consists of two different regions: one for headers, and the other for payload data. The memory used to store packets will never be released by NS-2 until the simulation ends. It repeatedly uses the assigned packets; all headers specified for a protocol are contained in each header region. This is despite the particular packet not using this particular header; only when the packet allocation is reused. The NS-2 can simulate a wide range of protocols, including TCP, UDP, FTP, HTTP, and various encryption algorithms such as the Diffie-Hellman algorithm; which in this research study is integrated into hashing.

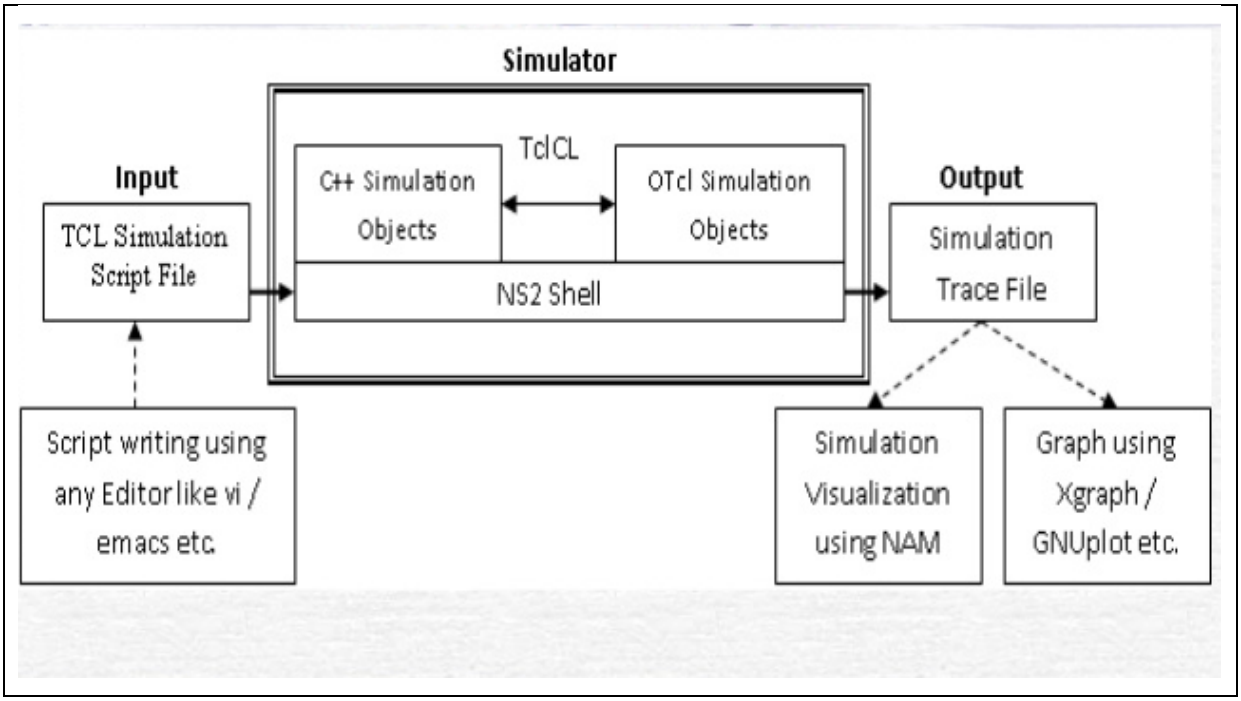

Figure 4.6: Network Simulator 2 architecture

4.5.2. Network Simulator 2 class pyramid

The Tcl Object class is at the top of the pyramid. Tcl Object is a superclass of all OTcl library objects, including network elements, timers, schedulers, and other objects, including NAM objects. The Ns Object class is the superclass of all the network component's basic objects that deal with packets, such as network devices and links. Based on the number of possible output data paths, the network components are divided into two sub-classes: connector and classifier. With only one output data path, network objects fall under the connector class; whereas switching artefacts with multiple output data routes is part of the classifier class.

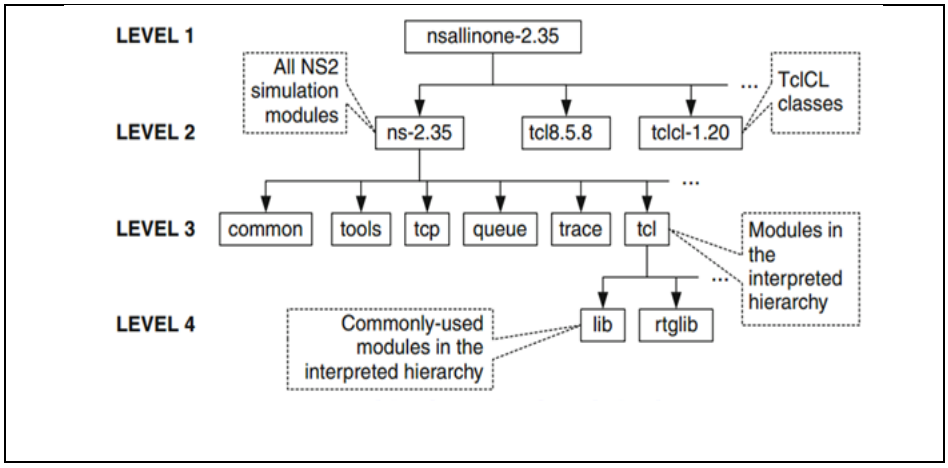

Figure 4.7: Network Simulator 2 class pyramid

4.5.3 Network Simulator 2 directory configuration

NS-2 is typically mounted in the ns-allinone-2.35 directory. The configuration of the NS-2 directory under the ns-allinone-2.35 directory is presented in Figure 4.8 below. In this research work, the C++ classes in NS-2 network components are implemented in the tcl subdirectory.

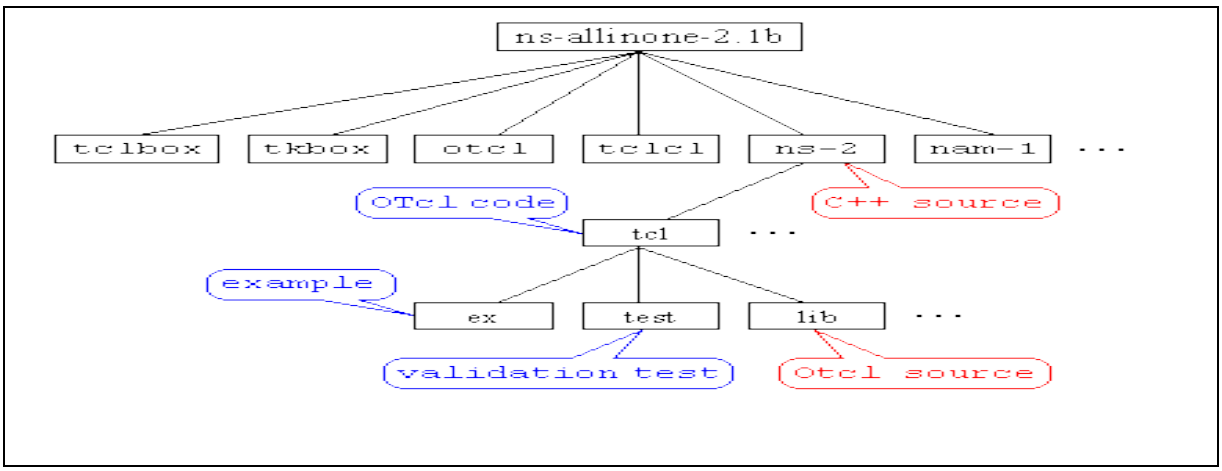

Figure 4.8: Network Simulator 2 directory configuration

4.5.4. Trace file output

The NS-2 trace file contains descriptions of the data transferred between the nodes on a network. During the simulation process of this research work, the NS-2 simulator automatically produces trace files. The trace format method specifies the trace file format produced by the class. Earlier simulator versions

were developed to maintain backward output file compatibility so that scripts continued to work after processing.

4.5.5. Adding encryption module in Network Simulator 2

Since NS-2 does not support any authentication or encryption algorithms, in this research study, a new security module or protocol was added to the NS-2. The security module facilitates in key sharing as well as in the implementation of encryption/decryption functions. This research study adds encryption and decryption functions for data fields in data packets to a new derived class. It also includes a key sharing function to ensure the confidentiality of data packets during transmission. A transpositional cipher and the Diffie-Hellman algorithm are the encryption/decryption and key exchange algorithms, respectively. The programming languages are C++, TCL, and AWK. NS-2 is extended by adding a new OTcl class that inherits built-in NS-2 classes. The OTcl script must be edited in order for the simulation to be realized. In the folder NS2.34/apps, a new packet class is developed. NS2.34/common/packet.h must be updated with the current packet name. This requires a change to the makefile to compile the newly added class. The new packet format must be declared at the TCL layer by editing the NS2.34/tclllib/ns-default.tcl file and inserting the name and default packet size value.

4.6. Cryptography Encryption Selection

As alluded to in Section 2.2, the major classification of cryptography algorithms is either symmetric or asymmetric. Symmetric key cryptography uses the same key for encryption and decryption; while in asymmetric key cryptography two different keys are used − one is for encryption and the another for decryption. Asymmetric cryptography consumes more power and memory due to the high computational process caused by using two keys which are private and public. Therefore, the focus area of this chapter is on asymmetric key cryptography for the analysis of their performance.

4.7. Simulation CESA

The CESA was proposed in this research study to reduce excessive power and memory consumption in order to efficiently secure public Wi-Fi networks. The simulation was conducted to validate and test the CESA. This section covers the specifics of configuring the NS-2 programme on the Linux operating system. In addition, the NS-2 simulation setting is discussed.

4.7.1. NS-2 Configurations

The NS-2 simulator can be installed on a variety of operating systems, including UNIX/Linux, Mac OS, and Windows. In this study, the NS-2 simulator was installed on a Linux operating system. The key reason for adopting Ubuntu for this study is that it is a free and widely used Linux operating system. The methods of setting up the NS-2 on Ubuntu are listed in Annexure A.

NAM is a simple Tcl/TK tool for network simulation that allows for inspecting traces and real-world packet tracks. NAM supports network topology network design and numerous analysis tools. The Wi-Fi simulation output for this research is shown in Figure 4.9 using the NAM console button in the next simulation scenario.

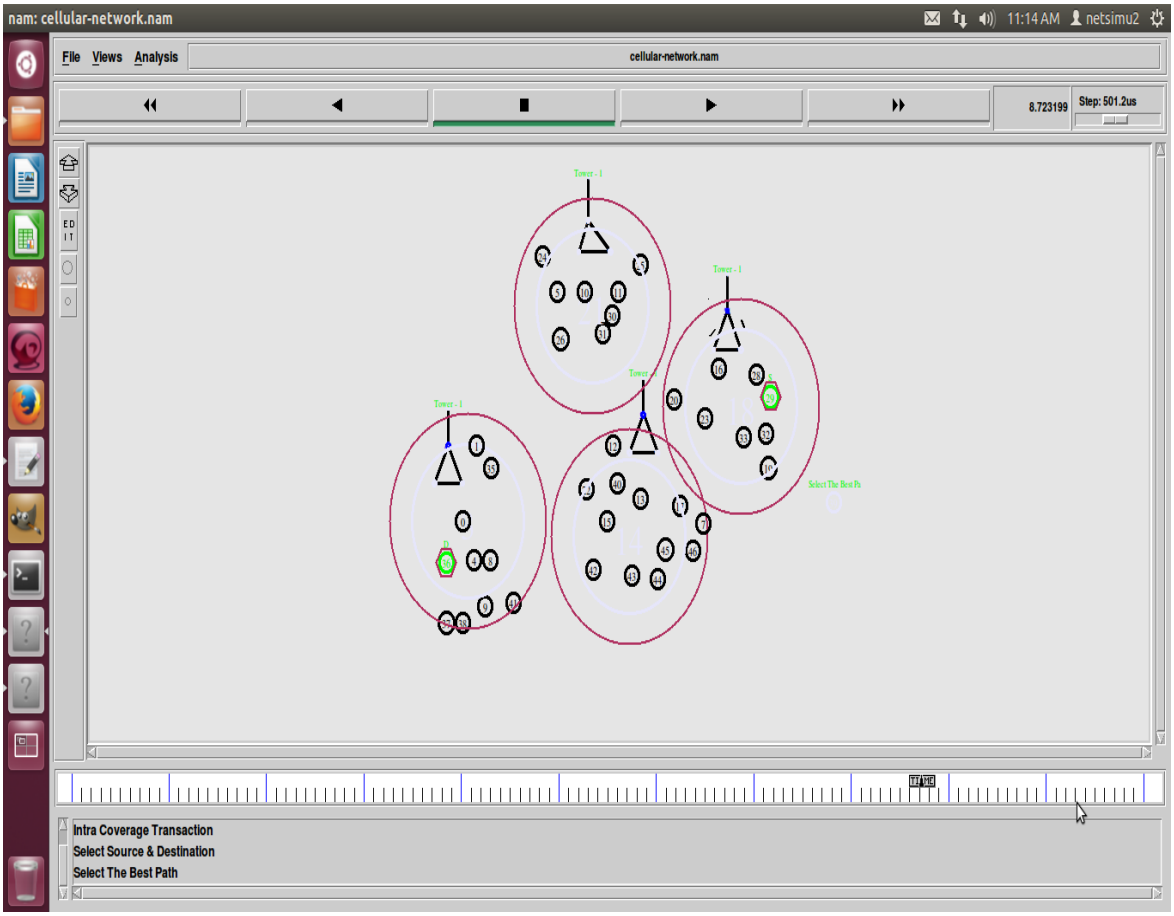

Figure 4.9: NAM simulation output for public WI-FI

4.7.2. Simulation methods

The proposed CESA was simulated in both ordinary and man-in-the-middle attack scenarios. The simulator scenarios are designed to show how the CESA performs both when there is a man-in-the-middle attack and when there is not.

In the second part of the simulation, a model is developed to detect and prevent man-in-the-middle attacks, as well as to reduce excessive power and memory consumption by employing a variety of mobile nodes, speeds, and traffic loads. Three different simulation scenarios were conducted. A man-in-the-middle attack was simulated in the second scenario.

The node-type scenario is created randomly, which means that the man-in-the-middle node addresses and the start time of the malicious behaviour are completely random. For simulation reasons, the start time of malicious node behaviour is randomly set between 0 and 50 seconds; whereas the start time of

data for transmission to a connection is set between 0 and 40 seconds independent of the node type.

This research study focuses solely on the results of TCP traffic. To model node mobility, the random waypoint method was used. Various node speeds (0 to 30 m/s) were used in this research study.

4.7.3. Simulation parameters

The Wi-Fi network performance is primarily determined by end-to-end connectivity; and the simulation scenario is designed to stimulate network security by increasing network throughput and packet transmission between nodes inside the scenario, utilizing cryptographic techniques. In this simulation, the CESA was employed to encrypt data packets sent between nodes.

First, the simulation process and results analysis are used to determine the architecture and configuration of nodes, as well as MAC layer properties for various address types, protocol types, channel types, simulation time, modulation type, idle, sleep power, and wireless transmission modes. The parameters of the simulation scenario are presented in Table 4.2.

Table IV: Simulation Parameters

Parameters	Values
1. Network area	500m x 500m
2.Number of nodes	40
3. Protocol type	TCP
4. Antenna model	Omnidirectional
5. Max package	50
6. Type of the MAC	802.11
7. Node speed	0-30

8. Transmission speed	1,3 Mbps
9. Bandwidth	30MHz

4.8. Summary

This chapter offered an overview of computer network simulation techniques. The background to simulation tools and software tools that can be utilized in public Wi-Fi networks is thoroughly discussed. To find the optimal tool for this study to employ in simulation, some open-source code simulators were evaluated. Finally, the simulation requirements for this study have been met. This chapter covered the simulation environment as well as the NS-2 architecture. The proposed CESA is examined and tested in the following chapter using NS-2 software.

Chapter 5: Results & Discussions

5.1. Introduction

As briefly discussed in Section 4.2, simulators have been used across the various areas of research to study and examine designed and implemented algorithms under varying network configurations. The purpose of these tools is to model the behaviour of real-time communication networks. The reason for using a network simulator is that it is time-consuming and very expensive to set up a real-time communication network. In this chapter, therefore, the experimental evaluations of the proposed CESA are thoroughly discussed. The experimental evaluations rely on the obtained results which were then analysed using AWK scripts. The study considered three performance metrics including shared key generation time, encryption time, and decryption time.

These metrics were chosen on the basis that this research was conducted to improve QoS in any wireless network by mitigating or reducing the success of man-in-the-middle attacks. From the simulations of the proposed CESA, the simulator generated trace files automatically; and those were used to collect and record the obtained results. Thereafter, the collected data were analysed to calculate key generation time, encryption time, and decryption time. The R Programming language was used to draw results in the form of graphs. The analysed results were obtained through multiple simulations which were conducted. R Programming language has been in normal use since 2014 as a data plotting programme. The language offers the creation of interactive graphs. It is currently maintained by the R Foundation for Statistical Computing. A sample R Programming code is shown in Appendix D. As mentioned earlier, AWK scripts were written to calculate the averages of the defined performance metrics.

The remainder of this chapter is structured as follows: In Section 5.2, the experimental evaluations of the proposed algorithm are presented. In Section 5.3, this chapter is summarized and concluded.

5.2. Experimental Evaluations

In Chapter 4, Figure 4.9 shows the simulation topology which was designed and implemented using TCL scripts in NS-2 in order to evaluate the performance of the proposed algorithm. The proposed algorithm was simulated to evaluate the benefits of integrating the Diffie-Hellman and hashing algorithms. To simulate the algorithm, NAM interface was used to observe packet losses, the locations of client machines and servers, and how the packets are transmitted. The NAM interface was used to evaluate the proposed algorithm against a modified Diffie-Hellman and AES, as shown in Figure 4.9. The network topology consists of 40 mobile nodes that represent client machines. Each of these machines is dynamically configured with a unique IP address. In addition to the client machines, the network also includes access points (APs) that are configured with a static IP addresses, in contrast to the dynamic configuration used for the client machines.

The CESA was implemented at the distribution layer represented by APs in the network. The role of APs is to centrally coordinate the network. As shown in Table 4.2, the simulations were performed using a propagation model of the omnidirectional model defined offered in the simulation environment.

The simulator was installed and executed on Linux Ubuntu 16.04.5 LTS. The OS was installed and run on an Oracle VM VirtualBox Manager Version 4.3.20 developed in 2014 by Oracle Corporation. The network topology was configured and implemented using C++ and OTcl code. The simulation was run several times to ensure quality and reliable results. Unlike other tools, the advantage of NS-2 is that it automatically records the results including packet

transmissions, loss, and much more. As discussed previously, R Programming was employed to graphically display the obtained and analysed results. Meanwhile, AWK scripts helped with the calculation in terms of results analysis.

5.3. Performance Metrics

The proposed model was examined against enhanced Diffie-Hellman (EDH) and AES. The motivation behind choosing these two encryption algorithms is that both were proposed to reduce the success of man-in-the-middle attacks in a computer network. EDH was proposed to facilitate the secure exchanging of cryptographic keys over a public channel. Meanwhile, AES facilitates the encryption of data shared or communicated over the network; and is commonly used for classified information by the United States (US).

In general, encryption is the process of encoding information. This process converts the original representation of the information, known as plaintext, into an alternative form known as ciphertext. Ideally, only authorized parties can translate a ciphertext back to plaintext, thus accessing the original information. Encryption does not itself prevent interference, but denies the intelligible content to a would-be interceptor.

However, AES uses too simple an algebraic structure; and each block is always encrypted in the same way. Meanwhile, it is too complex to implement in software taking both performance and security into consideration. The biggest weakness of EDH is that it does not establish the identity of the other party, making it vulnerable to man-in-the-middle attacks. This means that it does not authenticate any party in the transmission. Nevertheless, unlike EDH, AES is less open to attacks. Consequently, in this study, Diffie-Hellman has been integrated with a hashing function which includes the process of encryption and decryption applied. Such efficiently curbs out or reduces the success of man-in-

the-middle attacks in public Wi-Fi networks. This has improved the QoS in free public wireless networks.

As mentioned previously, the performance metrics considered during the evaluation of the proposed CESA against EDH and AES are as follows:

[1] **Key Generation Time** – the measurement of the length of time it takes to generate the security key within a network.
[2] **Encryption Time** – the measurement of the length of time it takes to change the data to an unreadable format within a network.
[3] **Decryption Time** – the measurement of the length of time it takes to change the data from unreadable to readable format within a network.

5.3.1. Key generation time

As mentioned in Chapter 3, the proposed algorithm modifies the Diffie-Hellman algorithm by applying hashing to improve its security. This is because the Diffie–Hellman algorithm is vulnerable to man-in-the-middle attacks; and thus, the hashing algorithm aids in reducing and preventing these attacks, by enforcing encryption and decryption of data between origin and destination. For the purpose of calculating the average key generation time, and other metrics, collected data from trace files was analysed through written AWK scripts given in Appendix C. The proposed algorithm improved on the key generation time of EDH and AES algorithms, as shown in Figure 5.1. This was achieved by modifying the Diffie-Hellman algorithm to quickly define or generate the key and let it be exchanged, so that the communication of data can be securely achieved. Meanwhile, hashing prevents or reduces the success of man-in-the-middle attacks by hashing the data for authentication, allowing for comparing and verifying that it is not modified, tampered with, or corrupted. This further ensures that those authentication keys are generated quickly and without being exposed to security vulnerabilities during the whole process. With CESA, it

took up to 59 seconds to generate the key; meanwhile, EDH and AES algorithms took almost 90 seconds, respectively.

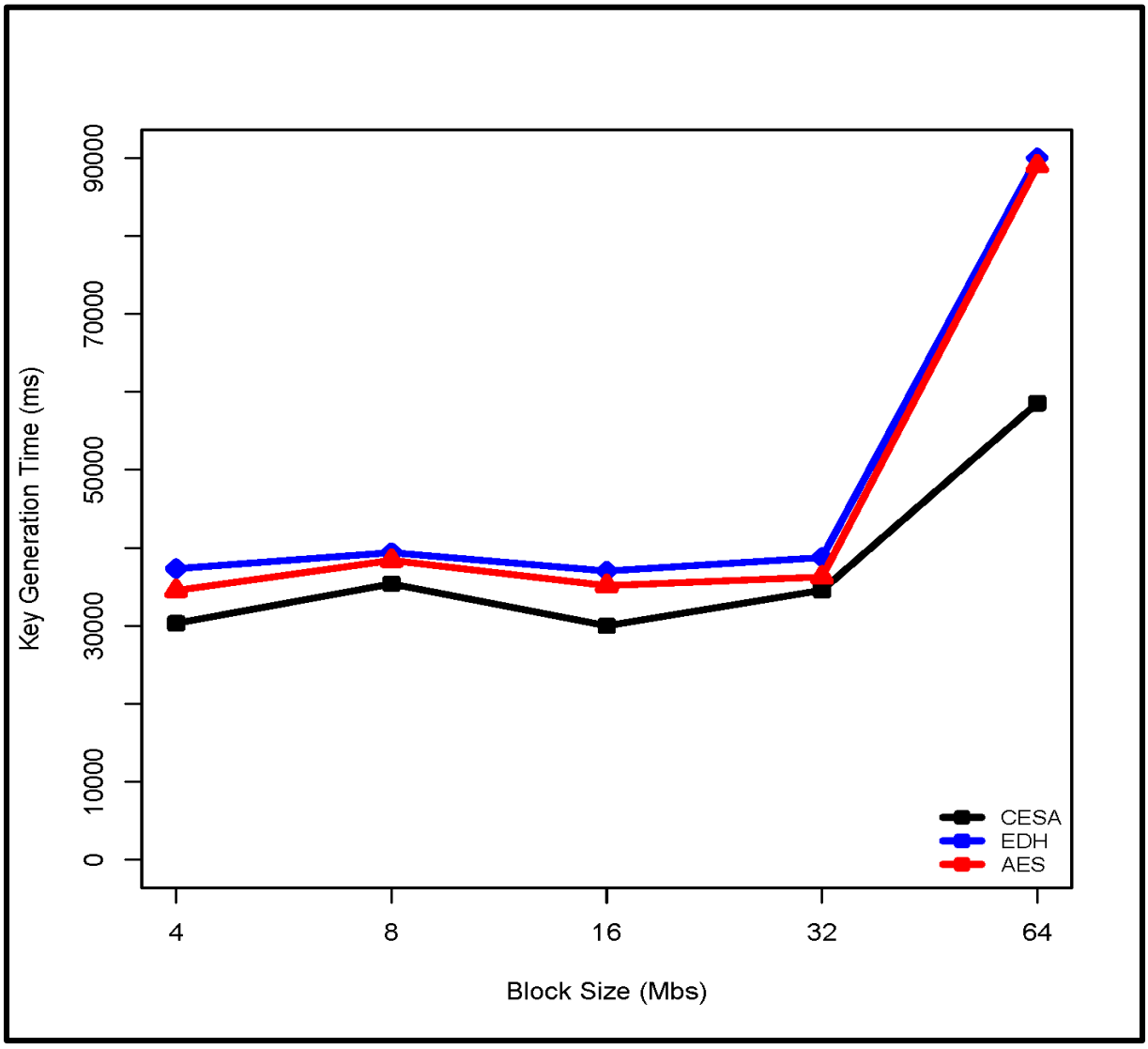

Figure 5.1: Key generation time

5.3.2. Encryption time

This study discusses the importance of the time it takes to encrypt the data shared on the public network. We present how the proposed algorithm performed against the EDH and AES algorithms. Figure 5.2 shows that the proposed algorithm reduces the time it takes to encrypt the data more than do the other algorithms. This has been achieved through hashing. This is much less complicated than AES in dealing with the process of encryption; better than relying on the Diffie-Hellman algorithm, which has been noted for its greater

vulnerability to man-in-the-middle attacks. The reduced encryption time aids in ensuring that attackers are not able rapidly to access the shared data before it is encrypted. This is because hashing makes it nearly impossible to guess the length of the hash should someone try to crack the shared data. Therefore, even if those attackers steal the data, they will have to find ways of decrypting it. Such is a very complicated task and normally is almost impossible to achieve. With CESA, it took about 98 seconds to encrypt the data; meanwhile, EDH and AES algorithms took almost 167 seconds to achieve the same result. This is because of the key-generating time that is reduced by the proposed algorithm, unlike with the conventional algorithms.

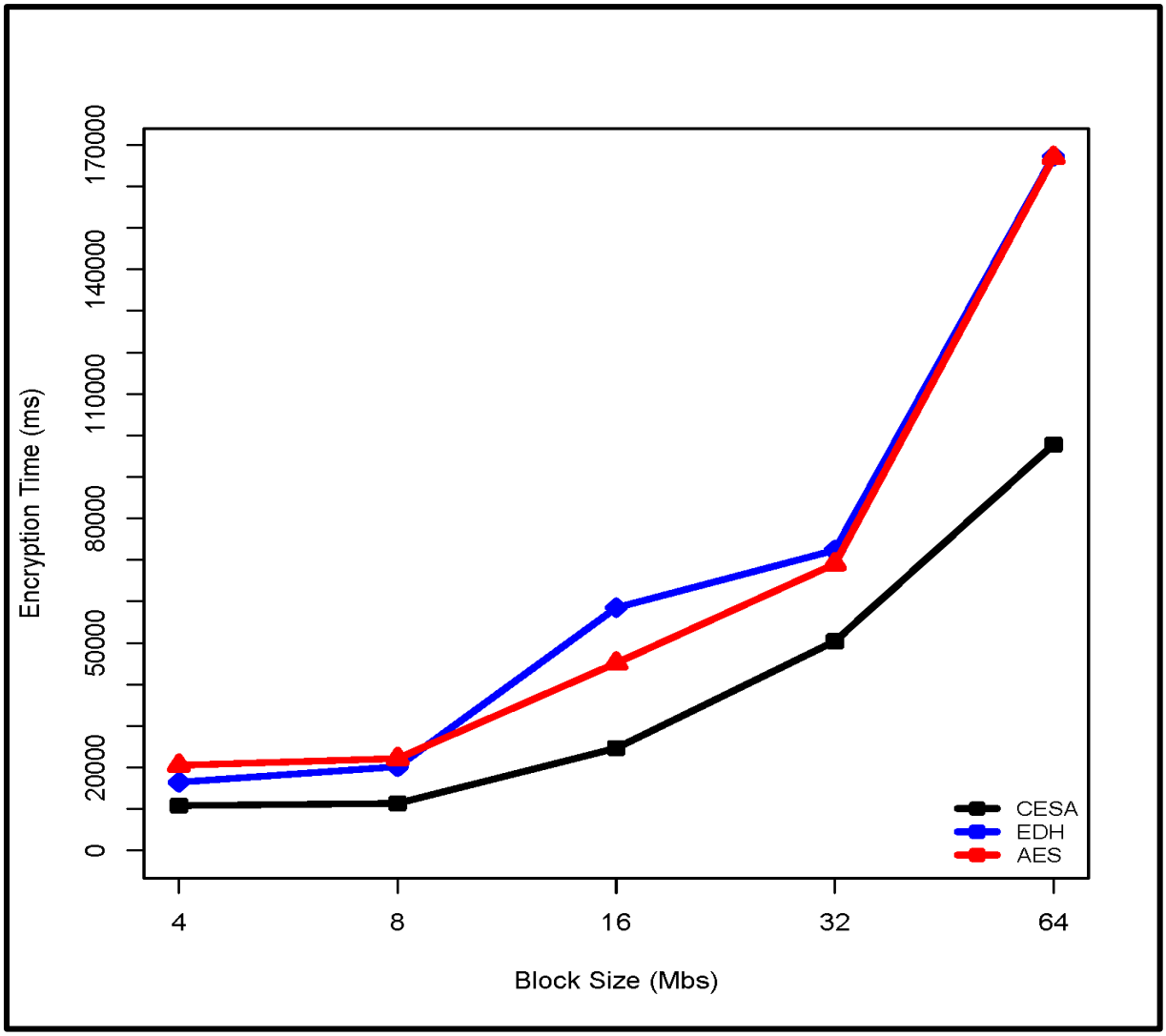

Figure 5.2: Encryption time

5.3.3. Decryption time

Decryption time, which is normally defined as the conversion of encrypted data to its original form, was also improved. Figure 5.3 reflects that decryption time has been well improved by the CESA compared with the EDH and AES algorithms. The reason behind these promising improvements is that the proposed CESA uses different key 128/192/256 bits to encrypt and decrypt data in blocks of 128 bits. With CESA, it took about 80 seconds to decrypt the data; meanwhile, EDH and AES algorithms took almost 160 seconds to gain the same result. The proposed CESA is thus seen as more robust against attacks and therefore very prompt in dealing with the processes of encryption and decryption.

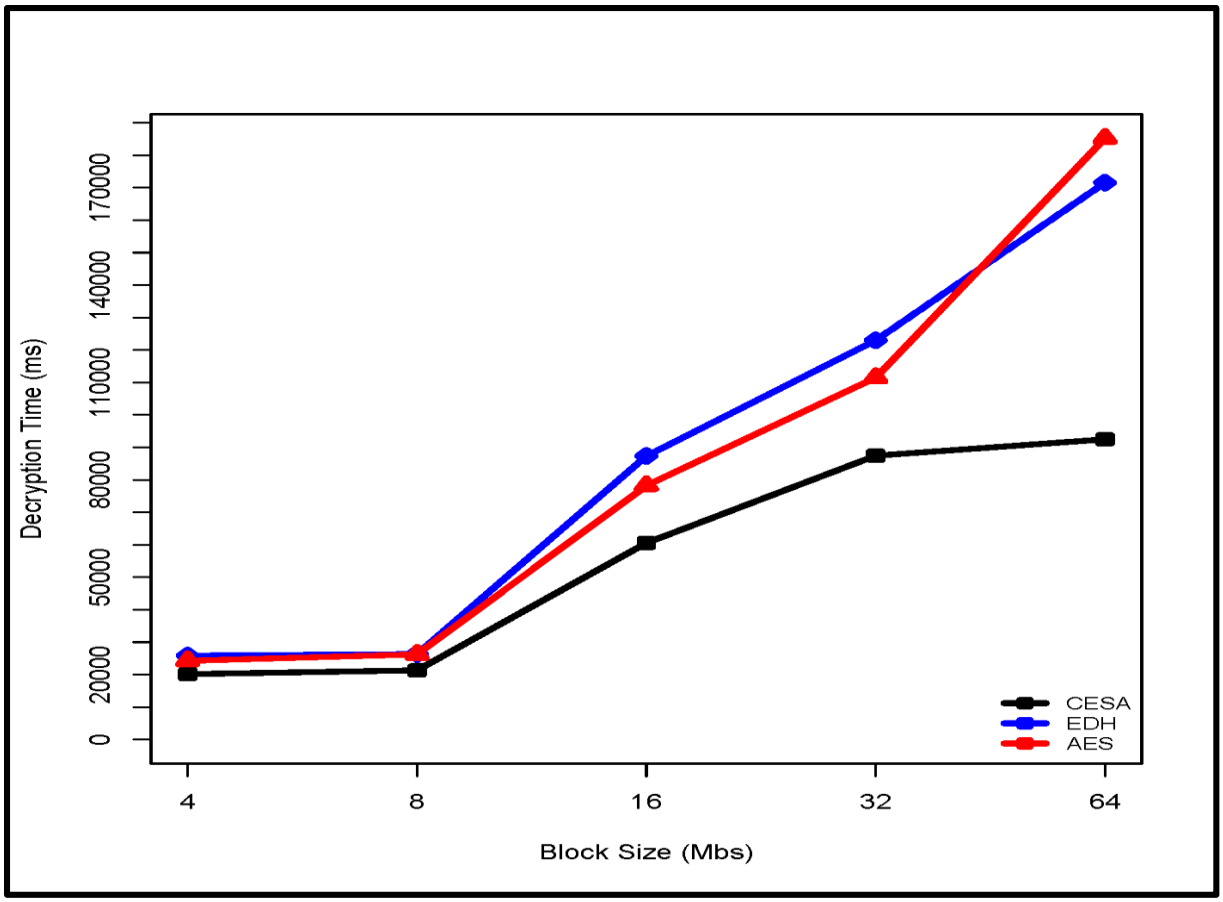

Figure 5.3: Decryption time

5.4. Summary

In this chapter, experimental evaluations of the proposed CESA algorithm were presented and thoroughly discussed. We provided discussions with regard to the improvements offered by the designed and implemented CESA in optimizing the performance and QoS by preventing or reducing the success of man-in-the-middle attacks. The results of the NS-2 through auto-created trace files were analysed and discussed in terms of the performance metrics considered in this study. The metrics included key generation time, encryption time, and decryption time. These metrics were considered in reality to prevent man-in-the-middle attacks, being rapid in the processing. Moreover, we discussed the reason that EDH and AES were chosen to be evaluated against the proposed CESA. We also discussed that R Programming language was used to graphically display the obtained results. In the next chapter, the conclusion and future work of this study are presented.

Chapter 6: Conclusion and Future Work

6.1 Introduction

In this research study, a CESA was designed by integrating a hashing algorithm with the Diffie-Hellman algorithm to prevent the man-in-the-middle attacks. The Diffie-Hellman algorithm is vulnerable to man-in-the-middle attacks; the hashing algorithm aims to reduce and prevent those attacks. Furthermore, the hashing algorithm aims to generate authentication encrypted keys. The algorithm ensures that the process of encryption and decryption of generated keys is carefully considered in preventing man-in-the-middle attacks encountered by the Diffie-Hellman algorithm.

Chapter 6 is organized as follows: Section 6.2 describes how the technical goals were achieved. Section 6.3 provides an overall summary of the research study. Section 6.4 describes the challenges encountered during the investigation. Section 6.5 presents the future work. Section 6.6 concludes the chapter.

6.2 Achievements of Objectives

All objectives defined in Section 1.5 of Chapter 1 have been achieved in this research project.

1. To identify and analyse the existing encryption algorithms used for free public wireless networks.

In Section 2.2, the existing encryption algorithms used for public wireless networks were identified and analysed in the literature review. The RSA, the Diffie-Hellman algorithm, the DSA and ECC using asymmetric techniques, were discussed. Encryption algorithms for wireless networks can use Bluetooth, WI-FI, WiMAX and LTE Wi-Fi technologies that are currently accessible on the market to ensure the security of the wireless applications. The drawback identified for Bluetooth technology was that it has a short-range and low-power wireless networking standard. Such standards have been widely implemented in

computing and communications devices as well as in peripherals, such as cellphones, keyboards, and audio headsets; making it subject to many wireless security threats − Bluetooth may easily become compromised. For protection, diverse security features and protocols have been introduced for Bluetooth in order to prevent potential serious attacks.

Wi-Fi networks mainly based on the IEEE 802.11 b/g standards have been explosively expanding. The most common security protocols in Wi-Fi are referred to as WEP and WPA. WEP was proposed in 1999 as a security measure for Wi-Fi networks to make wireless data transmissions as secure as in traditional wired networks. However, WEP is a relatively weak security protocol, having numerous flaws. Hence, it can be "cracked" in a few minutes using a basic laptop computer. As an alternative, in 2003, WPA was put forward for replacing WEP, while the improved WPA2 constitutes an upgraded version of the WPA standard. Typically, WPA and WPA2 are more secure than WEP; and thus they are widely used in modern Wi-Fi networks.

The computation of these authentication algorithms consumes a substantial amount of computing resources such as memory and battery power. The consumption of power may affect the device, switching off while the session is running. This may lead to attacks such as session hijacking attacks, KRACK attacks, and man-in-the-middle attacks. An attack is considered an active eavesdropping assault, mostly carried out by ARP cache poisoning on the MAC layer. An attack aims to spoof the MAC addresses of the legitimate nodes, claiming to be the intended sender or receiver of traffic between two legitimate nodes by intercepting a link between two legitimate hosts on the network. Once the legitimate node is battery-exhausted due to battery consumption caused by higher computation, the attacker may attempt to take over the session.

2. To design an encryption algorithm that reduces the high consumption of power and memory to efficiently secure data over free public wireless networks.

In Chapter 3, a discussion was provided on how a CESA was designed. A hashing algorithm and the Diffie-Hellman algorithm were integrated to prevent the man-in-the-middle attacks. In this research study, the keys generated during the encryption and decryption processes were used to prevent man-in-the-middle attacks. In Chapter 4, CESA was implemented using a NS-2 simulator.

3. To measure the performance of the proposed encryption algorithm against existing algorithms.

The performance of a CESA was evaluated based on key generation, encryption time and decryption time, to measure the power and memory consumption, thus securing public Wi-Fi networks. The results from the NS-2 through auto-created trace files were analysed and discussed in terms of the performance metrics considered in this study. The proposed algorithm was designed to improve on the Diffie-Hellman algorithm which is vulnerable to man-in-the-middle attacks, by generating keys between communicating devices. The performance metrics included key generation time, encryption time, and decryption time. These were considered by the reality that the sooner these are processed the sooner man-in-the-middle attacks can be prevented.

6.3 Summary of Research

The proposed CESA prevents the man-in-the-middle attacks by generating a key between communicating devices. The proposed algorithm improved on the Diffie-Hellman algorithm because this algorithm is vulnerable to man-in-the-middle attacks. In addition, the proposed CESA reduced computational time; thus it consumes less memory and power. The proposed algorithm was implemented using a NS-2 simulator.

6.4 Encountered Challenges

The NS-2 runs on two different programming languages − C++ and OTcl. C++ is used for reverse ends and OTcl is used as a front end for actual simulation.

Hence, implementing the proposed algorithm using NS-2 was a challenging task. The other challenge faced was designing the proposed algorithm, and learning NS-2 languages.

6.5 Future Work

The proposed CESA will be evaluated in a more complex free public wireless network with more network users. In addition, the proposed CESA will be evaluated in a real free public wireless network in order to mitigate man-in-the-middle attacks and computational time. Such includes key-generation time, encryption time, and decryption time.

6.6 Summary

In this chapter, it was concluded that the proposed CESA was able to prevent man-in-the-middle attacks without increasing the encryption and decryption times. Thus, the proposed CESA reduced power consumption and memory. Having achieved all the defined goals, it is considered that the ultimate purpose of this research study has been successfully achieved.

References

1. Alanazi, H.O., Zaidan, B.B., Zaidan, A.A., Jalab, H.A., Shabbir,M. and AL-NabhanI,Y. (2012). "New Comparative Study Between DES, 3DES and AES within Nine Factors", Journal of Computing, pp. 152-157.
2. Arora, P., Singh, A and Tiyagi, H. (2012). "Evaluation and Comparison of Security Issues on Cloud Computing Environment", World of Computer Science and Information Technology Journal (WCSIT*)*, pp. 179-183.
3. Arora, P., Singh, A. and Tiyagi, H. (2012). "Evaluation and Comparison of Security Issues on Cloud Computing Environment", World of Computer Science and Information Technology Journal (WCSIT), pp. 179-183.
4. Bhagyavati, B, (2015). Wireless Security Techniques: An Overview. *In Wireless Security Techniques*. Columbus State University, September. Columbus State University: Bhagyavati, pp. 87-96.
5. Bhanot, R and Hans, R. (2015). A Review and Comparative Analysis of Various Encryption Algorithms. *International Journal of Security and Its Applications*, pp. 290-305.
6. Chowdhury, J.Z., Pishva, D. and NishanthA, D.G.G. (2010). "AES and Confidentiality from the Inside Out", the 12th International Conference on Advanced Communication Technology (ICACT), pp. 1587-1591.
7. Elminaam, D.S.A., KadeR, H.M.A. and Mohie Mohamed Hadhoud, M.M. (2014) "Performance Evaluation of Symmetric Encryption Algorithms", IJCSNS International Journal of Computer Science and Network Security, pp. 280-286.
8. Jiang, Y. and Chen, S. (2019). "A Novel Physical-layer Security Scheme for Internet of Things", *Journal of Physics: Conference Series*, pp. 42-90.

9. Kakkar, A., Singh, M.L and Bansal, P.K. (2012). "Comparison of Various Encryption Algorithms and Techniques for Secured Data Communication in Multinode Network", International Journal of Engineering and Technology, pp. 87-92.
10. Kester, Q.A. (2016). “A Hybrid Cryptosystem Based on Vigenere Cipher and Columnar Transposition Cipher”, International Journal of Advanced Technology & Engineering Research (IJATER), Issue 1, pp. 141-147.
11. Kumar, A., Jakhar, S and MAKKAr, S. (2012). "Comparative Analysis between DES and RSA Algorithm’s", International Journal of Advanced Research in Computer Science and Software Engineering, pp. 386-391.
12. Li, X., Dai, H.N., Wang, H and Xiao, H. (2016). "On Performance Analysis of Protective Jamming Schemes in Wireless Sensor Networks", *Sensors*, pp. 1987-1995.
13. Liang, Q. and Modiano, E. (2019). "Minimizing Queue Length Regret Under Adversarial Network Models", *ACM SIGMETRICS Performance Evaluation Review*, pp. 31-38.
14. Mandal, A.K., Parakash, C andTiwari, A. (2012). “Performance Evaluation of Cryptographic Algorithms: DES and AES”, IEEE Students’ Conference on Electrical, Electronics and Computer Science, pp. 1-5.
15. Mohammed, M., Mohammed, H., Ghalwash, M.A. and Mohammed, A.R. (2019). "Design and implementation of new security architecture for wireless network", *IOP Conference Series: Materials Science and Engineering*, pp. 012065-012071.
16. Nie, T.,Song, C and Zhi, X. (2010). "Performance Evaluation of DES and Blowfish Algorithms", IEEE International Conference on Biomedical Engineering and Computer Science (ICBECS- 2010), pp. 1-25.
17. Padmapriya, A. and Subhasri, P. (2015). ” wireless networks: Reverse Caesar Cipher Algorithm to Increase Data Security”, International Journal of Engineering Trends and Technology (IJETT), pp. 1067-1071.

18. Pavithra, S and Ramadevi, E. (2012). "Performance Evaluation of Symmetric Algorithms", Journal of Global Research in Computer Science, pp. 43- 45.

19. Pu, C and Zhou, X. (2018). "Suppression Attack Against Multicast Protocol in Low Power and Lossy Networks: Analysis and Defenses", *Sensors*, vol. 18, pp. 3236-3242.

20. Senigagliesi, L., Baldi,M and Chiaraluce, F. (2017). "Semantic Security with Practical Transmission Schemes over Fading Wiretap Channels", *Entropy*, vol. 19, pp. 491-497.

21. Seth, S.M. and ishra,R (2011). “Comparative Analysis of Encryption Algorithms for Data Communication”, International Journal of Computer Science and Technology, pp. 292-294.

22. Singh,G., Singla,A. and Sandha,K.S. (2011). "Cryptography Algorithm Comparison for Security Enhancement in Wireless Intrusion Detection System", International Journal of Multidisciplinary Research, pp. 143-151.

23. Smith, B.R; Murphy, S.; & Garcia-Luna-Aceves, J.J. Security distance-vector routing protocols. Proc. Symposium Network and dist. System Security, Los Alamitos, CA, Feb. 1997, pp. 85-92.

24. Somani, U., Lakhani, K and Mundra, M. (2010). “Implementing Digital Signatures with RSA Encryption Algorithm to Enhance the Data Security of Cloud in Cloud Computing”. 1st International Conference on Parallel, Distributed and Grid Computing (PDGC), pp. 211-216.

25. Swati, S. (2012). “Wireless Network Security Protocols A Comparative Study”. International Journal of Emerging Technology and Advanced Engineering, [Online], vol. 2, pp. 350-360.

26. Tagwa, A.B.G. (2016). “A Comparative Study between WEP, WPA and WPA2 Security Algorithms”. International Journal of Science and Research (IJSR), vol. 4 (5), pp. 2390-2391.

27. Thanh, P.D., Hoan, T.N.K., Vu-Van, H. and Koo, I. (2019). "Efficient attack strategy for legitimate energy-powered eavesdropping in tactical cognitive radio networks". Wireless Networks, pp. 3605-3612.

28. Toma, A., Regazzoni,C., Marcenaro, L. and Gao, Y. (2019). *Handbook of Cognitive Radio*, pp. 1987-1995.

29. Wan, L., Zhang, G., Cui, M. and Lin, F. (2017). "Proactive Eavesdropping via Pilot Contamination and Jamming". *Wireless Personal Communications*, pp. 1958-1963.

30. Xiang, Z., Yang., Cai, Y., Cheng, Y., Wu, H and Wang, M. (2018). "Exploiting Uplink NOMA to Improve Sum Secrecy Throughput in IoT Networks". *Wireless Communications and Mobile Computing*, pp. 1-6.

31. Zhou, X., and Tang, X. (2011). "Research and Implementation of RSA Algorithm for Encryption and Decryption". the 6th International Forum on Strategic Technology, pp. 1118–1121.

32. Panse, T. and Kapoor, V. (2017). “An Integrated Scheme based on Triple DES, RSA and MD5 to Enhance the Secutity in Bluetooth Communiation”, International Journal of Computer Applications, vol. 50(7), pp. 123-130.

33. Shahin, O., Aissa, A., Fouad, Y., Al-Mahdi, H., and Alsmarah, M. (2020). “A New Method of Data Encryption based on One to One Functions”. International Journal on Advanced Science, Engineering and Information Technology. 10. 10.18517/ijaseit.10.3.10765, pp. 1675-1681.

34. Su, N., Zhang, Y., and Li, M., (2019). “Research on data encryption standard based on aes algorithm in internet of things environment”. In 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), pp. 2071-2075.

35. Kaur, M. and Kaur, J., (2017). “Data Encryption Using Different Techniques: A Review”. International Journal of Advanced Research in Computer Science, 8(4), pp. 1-7.

36. Rani, S. and Kaur, H., (2017). “Technical review on symmetric and asymmetric cryptography algorithms”. International Journal of Advanced Research in Computer Science, 8(4), pp. 189-197.

37. Arya, A. and Soni, S., (2018). A literature review on various recent steganography techniques. International Journal on Future Revolution in Computer Science & Communication Engineering, 4(1), pp.143-149.

38. Alsharif, M.H. and Nordin, R., (2017). Evolution towards fifth generation (5G) wireless networks: Current trends and challenges in the deployment of millimetre wave, massive MIMO, and small cells. Telecommunication Systems, 64(4), pp.617-637.

39. Alblwi, S. and Shujaee, K., (2017). “A survey on wireless security protocol WPA2”. In Proceedings of the International Conference on Security and Management (SAM), pp. 12-17.

40. Chen, J., Feng, Z., Wen, J.Y., Liu, B. and Sha, L., (2019). “A container-based dos attack-resilient control framework for real-time UAV systems”. In 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 1222-1227.

41. Badholia, A., Verma, V., Kalkal, S. and Kashyap, S.K., (2018). “New Wireless Network Protocol: WEP, WAP, WAP2”. International Journal of Satellite Communication & Remote Sensing, vol. 4(2), pp.14-19.
42. Delgado, O., Kechtban, L., Lugan, S. and Macq, B., (2020). “Passive and Active Wireless Device Secure Identification”. IEEE Access, pp.83312-83320.
43. Taha, M.S., Rahim, M.S.M., Lafta, S.A., Hashim, M.M. and Alzuabidi, H.M., (2019). “Combination of steganography and cryptography: A short survey”. In IOP conference series: materials science and engineering, pp 145-151.
44. Hayouni, H. and Hamdi, M., (2020). A novel energy-efficient encryption algorithm for secure data in WSNs. The Journal of Supercomputing, pp.1-24.
45. Bisht, N. and Singh, S., (2015). A comparative study of some symmetric and asymmetric key cryptography algorithms. International Journal of Innovative Research in Science, Engineering and Technology, vol. 4(3), pp.1028-1031.
46. Algazy, K., Biyashev, R., Kapalova, N., Babenko, L., Ishchukova, E. and Nyssanbayeva, S., (2019). “Investigation of the different implementations for the new cipher Qamal”. In Proceedings of the 12th International Conference on Security of Information and Networks, pp. 1-8.
47. Kessler, G.C., (2016). “An Overview of Cryptography”, pp. 1-10.

48. Chen, L., Chen, L., Jordan, S., Liu, Y.K., Moody, D., Peralta, R., Perlner, R. and Smith-Tone, D., 2016. Report on post-quantum cryptography . US Department of Commerce, National Institute of Standards and Technology.

49. Sadkhan, S.B. and Abdulraheem, F.H., (2016). "Security evaluation of cryptosystems used in cloud networks". In 2016 Al-Sadeq International Conference on Multidisciplinary in IT and Communication Science and Applications (AIC-MITCSA), pp. 1-6.
50. Advani, N., Rathod, C. and Gonsai, A.M., (2019). "Comparative study of various cryptographic algorithms used for text, image, and video". In Emerging Trends in Expert Applications and Security, pp. 393-399.
51. Shakhov, V. and Koo, I., (2018). "Depletion-of-battery attack: Specificity, modelling and analysis". Sensors, vol. 18(6), pp.1849-1857.
52. Arif, M., Wang, G., Bhuiyan, M.Z.A., Wang, T. and Chen, J., (2019). A survey on security attacks in VANETs: Communication, applications and challenges. Vehicular Communications, vol. 19, pp.100-179.
53. Blerina, Ã., Prodani, R. and Qafzezi, K., (2017). "Symmetric versus Asymmetric Cryptographic Techniques and Security Issues under Various Applications". *Global Journal of Computers & Technology*, vol. *6*(1), pp.317-322.
54. Hidayat, T.N. and Riadi, I., 2021. Optimation wireless security IEEE 802.1 X using the extensible authentication protocol-protected extensible authentication protocol (EAP-PEAP). *International Journal of Computer Applications*, *174*(11), pp.25-30.
55. Ghodke, S.S., 2016. Network Security: Attacks and Defense. *Innovation in IT*, *3*(1), pp.23-28.
56. Prasad, M. and Manoharan, R., 2017, January. A secure certificate based authentication to reduce overhead for heterogeneous wireless network.

In *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)* (pp. 1-5). IEEE.

57. Abo-Soliman, M.A. and Azer, M.A., 2018, July. Tunnel-Based EAP Effective Security Attacks WPA2 Enterprise Evaluation and Proposed Amendments. In *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)* (pp. 268-273). IEEE.

APPENDIX B: TCL SCRIPT

```
#===================================================================
#  The code to simulate the proposed algorithm
#===================================================================
#initialize the variables
set val(chan)         Channel/WirelessChannel      ;#Channel Type
set val(prop)         Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)        Phy/WirelessPhy                ;# network interface type
set val(mac)          Mac/802_11                       ;# MAC type
set val(ifq)          Queue/DropTail/PriQueue        ;# interface queue type
set val(ll)           LL                                    ;# link layer type
set val(ant)          Antenna/OmniAntenna           ;# antenna model
set val(ifqlen)       50                                  ;# max packet in ifq
set val(nn)           6                    ;# number of mobilenodes
set val(rp)           AODV                 ;# routing protocol
set val(x)            500   ; # in metres
set val(y)            500   ; # in metres

#creation of Simulator
set ns [new Simulator]

#creation of Trace and namfile
set tracefile [open wireless.tr w]
$ns trace-all $tracefile

#Creation of Network Animation file
set namfile [open wireless.nam w]
$ns namtrace-all-wireless $namfile $val(x) $val(y)

#create topography
set topo [new Topography]
$topo load_flatgrid $val(x) $val(y)

#GOD Creation - General Operations Director
create-god $val(nn)

set channel1 [new $val(chan)]
set channel2 [new $val(chan)]
set channel3 [new $val(chan)]

#configure the node
$ns node-config -adhocRouting $val(rp) \
  -llType $val(ll) \
  -macType $val(mac) \
  -ifqType $val(ifq) \
  -ifqLen $val(ifqlen) \
  -antType $val(ant) \
  -propType $val(prop) \
  -phyType $val(netif) \
  -topoInstance $topo \
  -agentTrace ON \
  -macTrace ON \
  -routerTrace ON \
  -movementTrace ON \
  -channel $channel1
```

```
set n0 [$ns node]
set n1 [$ns node]
set n2 [$ns node]
set n3 [$ns node]
set n4 [$ns node]
set n5 [$ns node]

$n0 random-motion 0
$n1 random-motion 0
$n2 random-motion 0
$n3 random-motion 0
$n4 random-motion 0
$n5 random-motion 0

$ns initial_node_pos $n0 20
$ns initial_node_pos $n1 20
$ns initial_node_pos $n2 20
$ns initial_node_pos $n3 20
$ns initial_node_pos $n4 20
$ns initial_node_pos $n5 50

#initial coordinates of the nodes
$n0 set X_ 10.0
$n0 set Y_ 20.0
$n0 set Z_ 0.0

$n1 set X_ 210.0
$n1 set Y_ 230.0
$n1 set Z_ 0.0

$n2 set X_ 100.0
$n2 set Y_ 200.0
$n2 set Z_ 0.0

$n3 set X_ 150.0
$n3 set Y_ 230.0
$n3 set Z_ 0.0

$n4 set X_ 430.0
$n4 set Y_ 320.0
$n4 set Z_ 0.0

$n5 set X_ 270.0
$n5 set Y_ 120.0
$n5 set Z_ 0.0
#Dont mention any values above than 500 because in this example, we use X and Y as 500,500

#mobility of the nodes
#At what Time? Which node? Where to? at What Speed?
$ns at 1.0 "$n1 setdest 490.0 340.0 25.0"
$ns at 1.0 "$n4 setdest 300.0 130.0 5.0"
$ns at 1.0 "$n5 setdest 190.0 440.0 15.0"
#the nodes can move any number of times at any location during the simulation (runtime)
$ns at 20.0 "$n5 setdest 100.0 200.0 30.0"
```

```
#creation of agents
set tcp [new Agent/TCP]
set sink [new Agent/TCPSink]
$ns attach-agent $n0 $tcp
$ns attach-agent $n5 $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 1.0 "$ftp start"

set udp [new Agent/UDP]
set null [new Agent/Null]
$ns attach-agent $n2 $udp
$ns attach-agent $n3 $null
$ns connect $udp $null
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$ns at 1.0 "$cbr start"

$ns at 30.0 "finish"

proc finish {} {
 global ns tracefile namfile
 $ns flush-trace
 close $tracefile
 close $namfile
 exit 0
}

puts "Starting Simulation"
$ns run
```

APPENDIX C: AWK SCRIPT

```
#===========================================================
#   The code to calculate the key generation time
#===========================================================
BEGIN {
   recvdSize = 0
   startTime = 400
   stopTime = 0
}

{
        event = $1
        time = $2
        node_id = $3
        pkt_size = $8
        level = $4

# Store start time
if (level == "AGT" && event == "s" && pkt_size >= 512) {
if (time < startTime) {
        startTime = time
        }
   }

# Update total packets generated per time
if (level == "AGT" && event == "r" && pkt_size >= 512) {
   if (time > stopTime) {
        stopTime = time
        }
   # Rip off the header
   #hdr_size = pkt_size % 512
   #pkt_size -= hdr_size
   # Store received packet's size
   recvdSize += pkt_size
   }
}

END {
   printf("Key Generation Time[kbps] = %.2f\t\t
StartTime=%.2f\tStopTime=%.2f\n",(recvdSize/(stopTime-
startTime))*(8/1000),startTime,stopTime)
}
```

APPENDIX D: R PROGRAMMING SCRIPT

```
#=========================================================
#   The code to display the key generation time
#=========================================================
str_data <- read.table("./KGT.dat", header=T, sep="\t")
max_y <- max(str_data)
#define colours to be used for CESA, EDH, and AES lines
plot_colors <- c("black", "blue", "red")
#draw the graph using y axis that ranges from 0 to max_y.
plot(str_data$CESA, type="o", pch=22, lwd=4, col=plot_colors[1], ylim=c(0, max_y), las = 3,
axes=FALSE, ann=FALSE)
#make x axis using 4 - 64 bits labels
axis(1, at=1:5, lab=c(4, 8, 16, 32, 64))
axis(2, at=10000*0:max_y)
#create box around plot
box()
# graph EDH with blue line
lines(str_data$EDH, type="o", pch=23, lwd=4, col=plot_colors[2])
# graph AES with red line
lines(str_data$AES, type="o", pch=24, lwd=4, col=plot_colors[3])
#label the x and y axes with black text
title(xlab= "Block Size (Mbs)", col.lab=rgb(0,0.0,0))
title(ylab= "Key Generation Time (ns)", col.lab=rgb(0,0.0,0))
# create a legend in the bottomright corner of the box
legend("bottomright", names(str_data), cex=0.8, col=plot_colors, lwd=4, pch=22, bty="n");
# turn on device driver to flush output to PDF
dev.on()
```