Improved Encryption Algorithm for Public Wireless Network

Christopher Khosa *, Topside Mathonsi, Deon du Plessis, and Tshimangadzo Tshilongamulenzhe

Department of Information Technology, Faculty of Information and Communication Technology, Tshwane University of Technology, Pretoria, South Africa
Email: khosachristopher@gmail.com (C.K.); mathonsite@tut.ac.za (T.M.); duplessisdp@tut.ac.za (D.D); tshilongamulenzhetm@tut.ac.za (T.T).
*Corresponding author

***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 Enhanced Diffie-Hellman (EDH) and Advanced Encryption Standard (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 s, while the EDH and AES algorithms took almost 90 s. 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 s, while EDH and AES algorithms took almost 160 s. 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**

I. Introduction

Wi-Fi (public wireless fidelity) networks are important to many businesses and customers. This is so that consumers can access the internet from a variety of locations, including airports, shopping centers, and academic institutions, to name a few. Due to the wide variety of wireless networks, it is necessary to secure transmitted data to guarantee data availability, integrity, and secrecy [1].

The Advanced Encryption Standard (AES), Triple Data Encryption Standard (3DES), and Data Encryption Standard (DES) are only a few of the cryptographic algorithms that have been presented today for Wi-Fi networks. These algorithms are appropriate for private Wi-Fi networks since they make use of public key exchange [1]. The most widely used encryption algorithms are DES, 3DES, and AES [2]. One key with 64 bits is used by DES. AES uses a variety of (128, 192, 256) bit keys, while 3DES uses three 64-bit keys. In Blowfish, several (32–448) bit keys are employed. The National Institute of Standards and Technology uses DES as its first encryption standard. Both the key and block sizes for DES are 64 bits.

The negatives of DES, however, are that it is vulnerable to brute-force attacks and uses a lot of memory, making it a risky block cipher [1, 2]. The improved version of DES is called 3DES. 3DES employs a 64-bit block size and a 192-bit key, compared to DES's 56-bit key and 64-bit block size. To increase the level of encryption and lengthen the typical period during which the encryption remains secure, the encryption mechanism employed in 3DES is applied three times consecutively. Moreover, numerous investigations have demonstrated that 3DES is less effective than other block encryption methods in terms of speed [1]. AES is a block cipher (sometimes referred to as Rijndael). It offers changeable key lengths of 128, 192, or 256 bits. Variable key lengths of 128, 192, or 256 bits are supported by AES. Depending on the length of the key used, it uses 10, 12, or 14 rounds of encryption and operates on 128-bit data blocks. AES seems to be quick and reliable, therefore it can be used on a variety of platforms, especially small ones. The high power and memory consumption of this method is a drawback.

An important foundational component of information security in public Wi-Fi is the use of cryptographic methods. However, a survey of the literature revealed that these algorithms frequently use a lot of power, memory, and CPU time [1, 2]. They are not appropriate for public Wi-Fi because they add resource overheads and processing

Manuscript received June 9, 2023; revised August 10, 2023; accepted September 1, 2023; published February 15, 2024.

complexity. As a result, a Computational Efficient Secure Algorithm (CESA) was created as part of this research study to protect data transmitted over a public Wi-Fi network. The CESA was developed by combining the Diffie-Hellman and hashing algorithms to prevent man-in-the-middle attacks. Diffie-Hellman computation is believed to be vulnerable to man-in-the-middle attacks. The suggested approach improves network performance, protects transmitted data, and uses less memory and power to secure zones over Wi-Fi networks. The suggested algorithm used asymmetric encryption as its method. This is so because asymmetric encryption employs two keys. While the public key is used to encrypt data, the private key is used to decrypt it. The asymmetric cryptography method is significantly more trustworthy than the symmetric method since it requires two keys.

II. Literature Review

Different cryptographic techniques are frequently used by public Wi-Fi networks to safeguard data sent for keeping sensitive information secure, encryption is a vital tool. The confidentiality of the information being communicated or stored is helped by encryption, which transforms plaintext into an unintelligible format (ciphertext). Encryption can offer additional security advantages in addition to confidentiality, including digital signatures, authentication, and secret key management [3]. To guarantee the information's confidentiality, integrity, and availability, encryption techniques are used. Additionally, this stops information from being duplicated and tampered with.

Over the network. The 3DES, DES, and AES algorithms are a few of the often-used ones. These public key exchange-based algorithms offer a high level of protection for confidential data sent over the network. Since a public exchange relies on a temporary, mathematical key developed to encrypt data delivered over unprotected internet channels, the public key uses a lot of memory and power [1–3].

A. Overview of Cryptographic Encryption Algorithms

The practice and study of information concealing, and verification is known as cryptography. When information is transferred through the internet or another medium, it is commonly referred to as the study of secrets. It is the technique of concealing data in ciphertext, a non-readable format, to prevent unauthorized access [3]. The message can only be translated into plaintext by those who have access to a secret key. According to the research of Taha *et al*. [3], cryptography is the use of protocols, algorithms, and other techniques to systematically prohibit or deny illegal access to sensitive data. Due to the intensive computational process involved in encrypting data, encryption operations use a lot of memory and power, which negatively affects network speed. "Plaintext" refers to the message that is sent directly from the sender to the recipient, whereas "ciphertext" refers to the message that is delivered through the channel. Encryption and decryption are the two steps of cryptography. Encryption is the conversion of plaintext into ciphertext. Decryption is the process of going backwards to turn ciphertext into plaintext. A secret key and an algorithm make up the encryption process.

Data that is encrypted is secured using this key. The algorithm will result in a specific output depending on the secret key utilized. The algorithm's output will change if the secret key is altered. Symmetric key algorithms and asymmetric key algorithms are the two subcategories of cryptography. In symmetric key cryptography, the same key is used for both encryption and decryption; in asymmetric key cryptography, however, two separate keys are used, one for encryption and the other for decryption [4].

1) Types of encryptions

Since asymmetric encryption is the subject of this research project, only asymmetric encryption will be covered in this section. The speed of asymmetric encryption is typically 1,000 times slower than that of symmetric encryption. Accordingly, processing asymmetric encryption or decryption may require a thousand times more CPU power than processing symmetric encryption or decryption [5]. The asymmetric key approach requires additional processing power and memory due to its high computational complexity [6]. Therefore, the goal of this project was to create asymmetric encryption that required less computational complexity to use less memory and power. The major encryption techniques' classification is shown in Fig. 1.

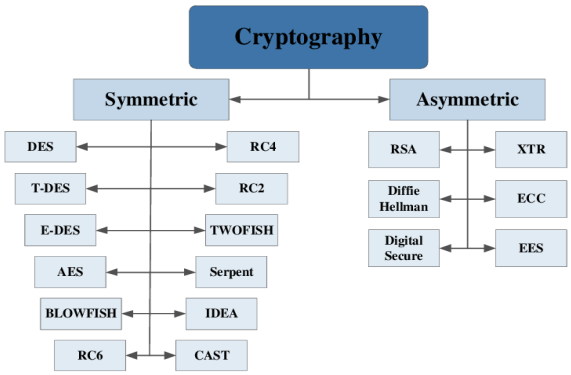

Fig. 1. Classification of encryption algorithms [6].

2) 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 [7]. 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 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.

B. 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 [8]. 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 [9]. This has led to many potential attacks such as brute-force attacks, man-in-the-middle attacks, timing attacks, and selected ciphertext attacks [8, 9].

C. Diffie-Hellman Algorithm

The Diffie-Hellman is one of the first public key processes: it is a way of safely exchanging cryptographic keys [8, 9]. 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 Denial-of-Service (DoS) attack [10]. Digital Signature Algorithm (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 [11]. Memory space, bandwidth, and power consumption are a constraint to the DSA process. Such can therefore lead to a session hijacking attack [12].

D. 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). The DSA is based on the exertion of computing discrete algorithm difficulties [13]. 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. Memory space, bandwidth, and power consumption are a constraint to the DSA process. Such can therefore lead to a session hijacking attack [11–13].

E. 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. 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 [14] (See Table I).

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

1) The contributions of the study

The proposed CESA algorithm was designed always to ensure secure data and reliability to all authorized individuals over free public wireless networks.

The proposed CESA algorithm allowed mobile devices on Wi-Fi to generate a shared hidden session key.

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.

2) Related works

Numerous studies have been conducted over the past few years on authentication methods for safeguarding wireless network users' personal information:

Yu and Kim [9] 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 encryption of the data. The experiment is secured towards timing and Power Supply Analysis (PSA) attacks. The CESA algorithm, as proposed, serves to thwart or minimize the efficacy of man-in-the-middle attacks. Additionally, it guarantees the swift and secure generation of authentication keys, mitigating potential vulnerabilities throughout the entire process. The algorithm's accelerated encryption time plays a crucial role in preventing attackers from swiftly accessing

unencrypted shared data. Furthermore, the proposed algorithm enhances decryption time, encompassing the conversion of encrypted data back to its original format.

Mandal and Dutta [10] compared the two most widely used symmetric encryption techniques namely DES and AES based on avalanche effect due to one-bit variation in plaintext keeping the key constant, avalanche effect due to one-bit variation in key keeping the plaintext constant, 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 cipher text. The Avalanche effect is very high for AES as compared to DES whereas memory requirement and simulation time for DES is greater than that of AES, which shows AES is better than DES in terms of power consumption.

Kumar *et al.* [15] proposed a DES system for data protection. The DES encryption method is a text data encryption method that uses a symmetric block cipher and consists of two steps: encryption and decryption. The valid length of the key used for DES encryption is 56 bits. Cryptography The DES encryption process consists of an 8-step process. The problem with this approach is that it is a complex and time-consuming process due to the length of the data being processed.

Elminaamn *et al.* [16] studied the performance of symmetric encryption algorithms. This white paper evaluates the six most used cryptographic algorithms (AES (Rijndael), DES, 3DES, RC2 and Blowfish, and RC6). Data block sizes, different data types, different battery consumption, different key sizes, and final encryption / decryption speed comparisons were performed with different settings for each algorithm. The following results were shown in the experimental simulation. If the results are displayed in either base hexadecimal or base64 encoding, there is no significant difference. It turns out that RC6 takes less time than all algorithms except Blowfish when the packet size is changed. We found that RC2, RC6, and Blowfish had a time disadvantage over other algorithms when changing data types such as images instead of text. By comparison, 3DES is still bad compared to the DES algorithm. Finally, when changing the key size, it can be shown that the larger the key size, the greater the change in battery and time consumption (AES algorithm or RC6 only).

Pu and Zhou [17] proposed a performance evaluation of common cryptographic algorithms for the throughput and power consumption of wireless systems and evaluated the efficiency of the selected symmetric key algorithm. AES, DES, and BLOWFISH were the algorithms of choice. The quality and power consumption of wireless devices are evaluated, and performance evaluations, various text, audio, and image files are used. The results showed that BLOWFISH performed better than AES and DES. This method has some drawbacks, such as weak keys and insensitivity to plain images. The proposed CESA algorithm reduced power consumption.

Pronika and Tyagi [18] compared the performance evaluation of various cryptographic algorithms. Based on parameters taken as time various cryptographic algorithms are evaluated on different video files. Different video files have different processing speeds at which various sizes of files are processed. Calculation of time for encryption and decryption in different video file formats such as Blob and .DAT, having file size from 1MB to 1100MB. Results showed that the AES algorithm is executed in lesser processing time and more throughput level as compared to DES and BLOW FISH.

Arora *et al*. [19] studied the performance of different security algorithms on a cloud network and on a single processor for different input sizes. Their aims are to find in quantitative terms like Speed-Up Ratio the benefits of using cloud resources for implementing security algorithms (RSA, MD5, and AES) which are used by businesses to encrypt large volumes of data. Three different kinds of algorithms are used, RSA (an Asymmetric encryption algorithm), MD5 (a hashing algorithm), and AES (a symmetric encryption algorithm). On different video files. Different video files have different processing speeds on which various sizes of the file are processed. Calculation of time for encryption and decryption in a different video file format such as .vob and .DAT, having file size from 1MB to 1100MB. Results showed that the AES algorithm is executed in lesser processing time and more throughput level as compared to DES and BLOW FISH. The block size is the basic unit of data that can be encrypted or decrypted in one operation. A larger Block size means greater security, all other factors being equal, but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion. Generally, a block size of 64 bit has been considered a reasonable tradeoff and was nearly universal in block cipher design. Block size used for DES, 3DES, and Blowfish is the same, 64 bits. 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.

In Ref. [20], a key management protocol centered around digital signatures was introduced to ensure secure data transmission. This process effectively minimizes the risk of false data injection and node failure, bolstering security for wireless internet-based data transfers. Through authentication of users and the safeguarding of transmitted data, any intercepted portion remains unintelligible due to encryption. However, this approach comes at the cost of doubling computational time, leading to significant power consumption on devices. Notably, this method bears resemblance to the CESA algorithm's data encryption aspect. However, the mentioned algorithm requires more time for data encryption in comparison. Conversely, the proposed CESA algorithm distinguishes itself by substantially expediting the encryption process, a pivotal deterrent against rapid unauthorized access to unencrypted shared data. Furthermore, the algorithm's enhancement of decryption time ensures seamless conversion of encrypted data into its original format.

In Ref. [21], a hybrid methodology was introduced. In this approach, plaintext blocks are encrypted utilizing both the Advanced Encryption Standard (AES) and Elliptic Curve Encryption (ECC) algorithms. Subsequently, data

compression technology is employed to obtain blocks of ciphertext. The ECC-encrypted MAC address and AES key are then combined to create a comprehensive ciphertext message. This hybrid approach demonstrated benefits such as decreased encryption time and heightened security, leading to reduced power consumption. However, it's worth noting that this technique is susceptible to Session Hijacking Attacks. Addressing these concerns, the proposed CESA algorithm places significant emphasis on countering hijacking attacks.

3) An enhanced encryption algorithm system design

The network architectural design and CESA implementation are presented in this section. This section starts off by going over the different system components as well as the building blocks for configuring these components. After that, by providing discussions on the physical connections configured on the network, this section fulfills the objectives of this study. This covers the significance of each system component, its configurations, and the advantages each component brings to the network to improve the Quality of Service (QoS) in open access WI-FI networks.

The goal of this study was to create a more advanced security system for open wi-fi networks that would enhance network efficiency. To reduce the effectiveness of man-in-the-middle attacks in these networks, the proposed approach was created. The proposed CESA design is mathematically presented in this section using exact mathematics.

The suggested design for it incorporates the layer 3 switch and Aps that connect to the server. The design also incorporates a gateway that is configured to serve as a link between the Aps and the internet, as seen in Fig. 1. The suggested design for it incorporates the layer 3 switch and Aps that connect to the server. The design also incorporates a gateway that is configured to serve as a link between the Aps and the internet, as seen in Fig. 1. The switch is configured to allow wireless connections from Aps because it performs transmission and hence operates at the distribution layer of the network. The MAC addresses of the connected devices to the switches serve as a means of identification. The transmitter helps with control and setup and supports a centralized network. The transmitter facilitates contact with the outside world and facilitates connections between Aps and the server. The layer 3 switch makes it easier for clients and service providers to communicate. It is capable of simultaneously performing bridging and routing duties. Additionally, it is simpler to configure and contains a lot of ethernet ports. It typically costs less than other switches and has the capacity to link more client's PCs.

The extensive QoS functions it offers further include packet prioritizing, classification, policing, tagging, queuing, and scheduling. To ensure a specific degree of QoS, the layer 3 switch is used in this work to encourage prioritizing. As a Domain Name System (DNS) address, the server is set up with a static IP address. To host databases, security settings, and archives, the server acts as a local cloud. On the other hand, Aps are set up with distinctive static IP addresses.

4) Wireless networks architecture design

As shown in Fig. 2 below, the proposed network architecture is mathematically modeled 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, $L = \{L_1, \dots, L_i, L_{i+1} \dots, L_K\}$ 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.

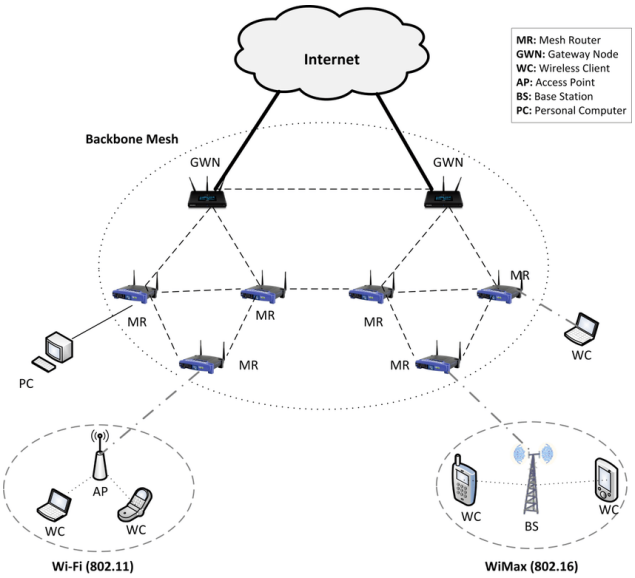

Fig. 2. Typical wireless networks architecture.

5) Encryption and decryption procedure

The encryption and decryption methods are shown in Algorithms 1 and 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 unable to be read because the attacker will never be able to know the key. Once the XNORed is finished, the obtained unreadable data is moved to the left. This will result in a new piece of unreadable text. The received text is then subjected to 1's complement. The last phase involves breaking the temporary ciphertext into two subsets, which then switch positions with each other. Thereafter, the final acquired ciphertext is translated back to ASCII values and sent to the destination.

Algorithm 1. Encryption Phases

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

Phase 2: Each letter is then converted to an ASCII decimal value (for example, a = "99"). Thereafter, it is translated to an 8-bit binary value (for example, "99" = 1100011).

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 side.

Phase 5: The performance from Phase 4 is multiplied by 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 are completed, the final obtained value in phase 6 is translated to ASCII decimal and then to an alphanumeric text.

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

The decryption method of the proposed algorithm is also covered in Algorithm 2.

Algorithm 2. Decryption Phases

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 is received. Once the computing of the decimal value is done, 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: The results of phase 4 are taken 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 are completed, the final obtained value is translated to ASCII decimal and then to the character.

F. Proposed Computation Efficient Secured Algorithm

In this study, CESA is proposed to reduce memory and power consumption and to reduce response time in public Wi-Fi networks. To obtain the objectives of this work, the hashing algorithm 3 is integrated with Diffie-Hellman algorithm to prevent attack by the man-in-the-middle. This is primarily because 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.

Algorithm 3. 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
Variable 1= XNORed (plaintext, key)
Variable 2= left swing respectively bit once in (Variable 1)
Variable 3=yield 1's compliment of (Variable 2)
Variable 4=split data into two parts in (Variable 3) & save as:
Set 0 = $\frac{m}{2}$ and set 1= $\frac{m}{2}$ to m
Ciphertext= exchange (Set0, Set1)
ELSE
Create key first, then go to phase 1
END IF
The process of decryption:
INPUT:
Key # Hashing key in Diffie–Hellman algorithm
Ciphertext $ binary values$
m = dimension of the letter
OUTPUT:
Variable 4=Ciphertext
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
Variable 2 = exchange (Set0, Set1)
Variable 1=yield 1's compliment of (Variable 2)
Variable = left shift each bit once in (temp1)
Plaintext= XNORed (plaintext, key)
ELSE
Create key first, then go to phase 1
END IF

G. Materials and Methods

Network simulators are valuable because they allow for the assessment of different solutions prior to actual implementation in real network infrastructure. In this research, the proposed algorithm that was developed in Section III was assessed using NS-2 to determine how well it performed. The NS-2 is considered as a discrete event simulator that is object-oriented and available in an open-source format. Therefore, it can be accessed for free on the internet.

In this research, the implementation of a CESA algorithm is presented. The proposed algorithm 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 securely 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).

The suggested CESA successfully prevent man-in-the-middle attacks in Diffie-Hellman algorithm by producing a key between the devices that are communicating with one another. Additionally, the proposed CESA encrypts and decrypts the generated key.

H. Network Simulators

Communication in wireless networks can be simulated using a variety of network simulators. This section discusses some of the most important network simulators and explains why NS-2 was chosen as the simulator for CESA algorithm. The simulators listed below were chosen because of their high level of popularity among researchers:

1) Objective modular network testbed (OMNeT++)

OMNeT++ is an open-source simulation instrument of discrete C++ language [22]. OMNeT+ + is not only used for network simulations, but 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 reasons, such as customizing the behaviour of the module, creating flexible topologies models (where parameters, and link structures may be defined), and shared 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 [12].

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

2) Network simulation 3 version

NS-3 is written in the C++ programming language. Unlike in NS-2, modern NS-3 anthology time hardware capabilities did not cause a challenge. As a result, NS-3 does not rely on Tool command language (OTcl) scripts to control the simulation; it can now be built entirely in C++. A modeling script, which is not possible in NS-2, can also be composed as a C++ program. 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 [23].

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 and 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.

3) 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 [24]. 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 optimisation [25].

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 the fact that basic MATLAB directives such as Linux, Microsoft Windows, and Mac systems need to be properly understood. Furthermore, sophisticated indulgent is also needed for features like 2D and 3D graphs, algebraic and differential equations, matrix calculations, and linear equations. MATLAB is not used in this research study as MATLAB has been designed mainly to solve complex numerical challenges [26].

4) 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 non-benefit groups to help improve the situation in the future. The simulation interface uses the OTcl and C++ for network topology and algorithm implementation, respectively [27].

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 like TCP and UDP; traffic sources like CBR Telnet, and FTP; and queue control mechanisms like Drop-Tail and algorithms for routing [27].

NS-2 is the most widely used open-source network simulator. It can investigate the behaviour of 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.

5) Simulating CESA

The primary objective of simulating the CESA algorithm was to determine whether the proposed algorithm improves wireless networks.

6) Simulation set-up

The CESA was proposed in this research study to reduce excessive power and memory consumption to efficiently secure public Wi-Fi networks. The simulation was carried out to validate and test the CESA. This section covers the specifics of configuring the NS-2 program on the Linux operating system. In addition, the NS-2 simulation setting is discussed [28–35].

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.

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 when there is no man-in-the-middle attack and when there is one.

In the second part of the simulation, a model is then 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 at random, which means that the man-in-the-middle node addresses and the start time of the malicious behavior are completely random. For simulation reasons, the start time of malicious node behavior is set between 0 and 50 s at random, whereas the start time of data for transmission to a connection is set between 0 and 40 s 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. Different node speeds (0 to 30 m/s) were used in this research study.

7) 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 II [28–35].

TABLE II. SIMULATION PARAMETERS

Parameters	Values
Network Area	500m × 500m
Number of nodes	40
Protocol type	TCP
Antenna Model	Omnidirectional
Max package	50
Type of the MAC	802.11
Node Speed	0–30
Transmission speed	1–3 Mbps
Bandwidth	30MHz

8) Experimental evaluation

The proposed algorithm was simulated to evaluate the benefits of integrating the Diffie-Hellman and hashing algorithm. 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. The topology is comprised of 40 mobile nodes to represent client machines each dynamically configured with a unique IP address. The networks also include Access Points (APs) statically configured with IP addresses, unlike 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. 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 by Oracle Corporation in 2014. 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 many more. As discussed previously, R Programming was employed to graphically display the obtained and analyzed results. Meanwhile, AWK scripts helped with the calculation in terms of results analysis.

III. RESULT AND DISCUSSION

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 decipher a ciphertext back to plaintext and access 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 doesn't 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 hashing function which includes the process of encryption and decryption applied to efficiently curb out or reduce the success of man-in-the-middle attacks in public Wi-Fi networks. This improved QoS in public-free wireless networks.

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

- Key Generation Time—the measurement of the amount of time it takes to generate the security key within a network.
- Encryption Time—the measurement of the amount of time it takes to change the data to an unreadable format within a network.
- Decryption Time—the measurement of the amount of time it takes to change the data from unreadable to readable format within a network.

A. Key Generation Time

As mentioned in Section III, the proposed algorithm modifies the Diffie-Hellman algorithm by applying hashing to improve its security. This is because Diffie-Hellman algorithm is vulnerable to man-in-the-middle attacks, and thus, hashing algorithm aids in reducing and preventing these attacks by enforcing encryption and decryption of data between origin and destination. For calculating the average key generation time, and other metrics, collected data from trace files was analyzed through written AWK scripts given in Appendix C. The proposed algorithm improved the key generation time as compared to EDH and AES algorithms as shown in Fig. 3. 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 done securely. Meanwhile, hashing prevents or reduces the success of man-in-the-middle attacks by hashing the data for authentication and to compare and verify 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 s to generate the key, meanwhile, EDH and AES algorithms took almost 90 seconds, respectively.

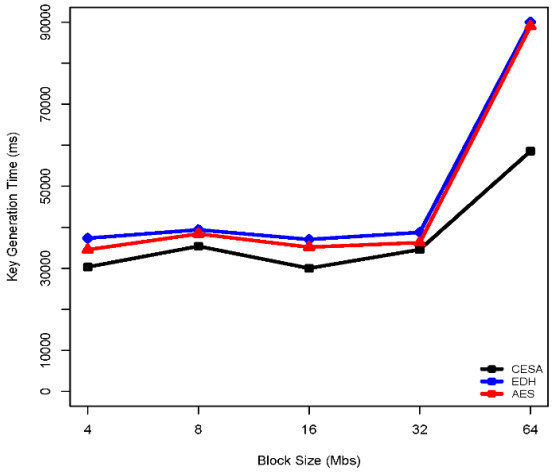

Fig. 3. Key generation time.

B. Encryption Time

Herein, 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 EDH and AES algorithms. Looking at Fig. 4, it shows that the proposed algorithm reduces the time it takes to encrypt the data compared to other algorithms. This has been achieved through hashing which is much less complicated than AES to deal with the process of encryption than relying on the Diffie-Hellman algorithm which has been noted that it is more vulnerable to man-in-the-middle attacks. The reduced encryption time aids in ensuring that attackers are not able to access the shared data quickly 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 can steal the data, they will have to find ways to decrypt it and that is a very complicated task and normally it is highly impossible in most cases. With CESA, it took about 98 s to encrypt the data, meanwhile, EDH and AES algorithms took almost 167 s, respectively. This is because of the key generating time that is reduced by the proposed algorithm compared to the conventional algorithms.

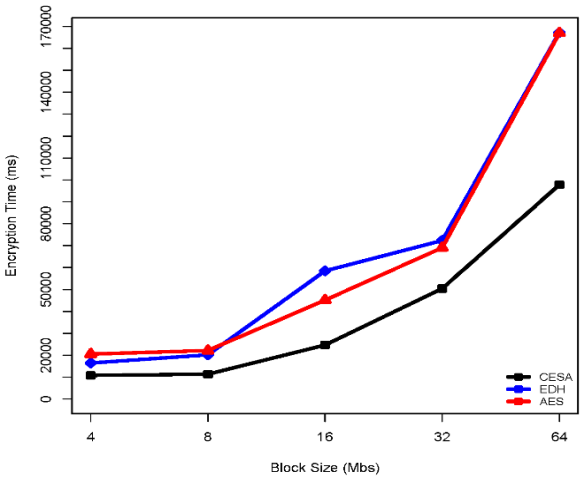

Fig. 4. Encryption time.

C. Decryption Time

Decryption time, which is normally defined as the conversion of encrypted data into its original form, was also improved. Looking at Fig. 5, decryption time has been well improved by the CESA as compared to EDH and AES algorithms. The reason behind these promising improvements is that the proposed CESA uses different keys 128/192/256 bits to encrypt and decrypt data in blocks of 128 bits. With CESA, it took about 80 s to decrypt the data, meanwhile, EDH and AES algorithms took almost 160 s, respectively. Hence, it makes the proposed CESA to be more robust against attacks and thus, it is very quick to deal with the processes of encryption and decryption promptly.

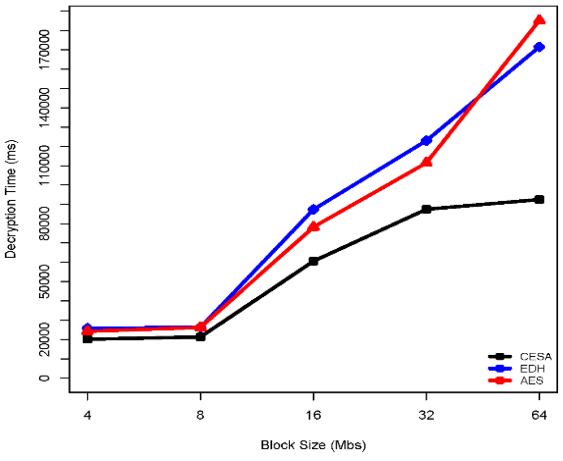

Fig. 5. Decryption time.

IV. Conclusion

This paper introduces the CESA algorithm to bolster the security of the susceptible Diffie-Hellman algorithm against man-in-the-middle attacks. The proposed CESA addresses this vulnerability by establishing secure key generation between communicating devices. The study also delves into encryption and decryption key generation, highlighting the algorithm's efficiency in terms of battery, memory, and CPU usage. While optimizing resources is commendable, scalability in intricate systems is crucial. The algorithm's effectiveness may dwindle with multiple devices or high-volume data transfers, risking performance bottlenecks and user experiences with network expansion. Regarding security, the paper emphasizes CESA's aim to counter man-in-the-middle attacks in Diffie–Hellman. Yet, the algorithm's security relies on accurate assumptions about potential threats. Misalignment with real-world risks could inadvertently expose vulnerabilities unforeseen during development. Ensuring security requires exhaustive threat analysis and rigorous assessment against advanced attacks beyond the paper's scope. Ultimately, CESA's reliability hinges on validation of its security mechanisms across practical and diverse scenarios.

Appendix A. 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 B. 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 C. 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()
```

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Christopher Khosa conducted the research and used a network simulator to implement the proposed solution. Christopher Khosa, Topside Mathonsi, Deon du Plessis, and Tshimangadzo Tshilongamulenzhe analyzed the data and wrote the paper. All authors had approved the final version.

FUNDING

This research study was funded by the Tshwane University of Technology.

ACKNOWLEDGMENT

I would like to express my sincere gratitude and appreciation for the generosity of my supervisors who contributed to the success of this study, DP du Plessis, TE Mathonsi and 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.

REFERENCES

[1] H. B. E. H. Hassan, "Comparative study of different cryptographic algorithms," *Journal of Information Security*, vol. 11, no. 11, 2020.

[2] M. Shahin, H. Vahid, and K. Mohammad, "EECA—Energy efficient congestion avoidance in wireless multimedia sensor network," in *Proc. 6th IEEE International Symposium on Telecommunications*, 2022, vol. 2.

[3] S. J. Mohammed and D. B. Taha, "From cloud computing security towards homomorphic encryption: A comprehensive review," *Telkomnika (Telecommunication Comput. Electron. Control*., vol. 9, no. 4, 2021
[4] H. Hayouni and M. Hamdi, "A novel energy-efficient encryption algorithm for secure data in WSNs," *The Journal of Supercomputing*, pp. 1–24, 2022.
[5] K. B. Logunleko, O. D. Adeniji, and A. M. Logunleko, "A comparative study of symmetric cryptography mechanism on DES, AES and EB64 for information security," *International Journal of Scientific Research in Computer Science and Engineering*, vol. 8, pp. 45–51, 2020.
[6] N. Bisht and S. Singh, "A comparative study of some symmetric and asymmetric key cryptography algorithms," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 4, no. 3, pp. 1028–1031, 2022.
[7] S. Yi and Y. Zhou, "Binary-block embedding for reversible data hiding in encrypted images," *Signal Process*., vol. 133, pp. 40–51, 2012.
[8] W. Stallings, "Digital signature algorithms," *Cryptologia*, vol. 37, pp. 311–327, 2013.
[9] H. Yu and Y. Kim, "New RSA encryption mechanism using one-time encryption keys and unpredictable bio-signal for wireless communication devices," *Electronics*, vol. 9, no. 2, 2020.
[10] M. Mandal and R. Dutta, "Efficient adaptively secure public-key trace and revoke from subset cover using framework," *Inscrypt*, vol. 11449, pp. 468–489, 2018.
[11] R. Kuang, M. Barbeau, and M. Perepechaenko, "Anew quantum safe multivariate polynomial public key cryptosystem over large prime galois fields," *Submitted to Scientific Reports Nature*, vol .3, 2021.
[12] M. E. Haii, M. Chamoun, A. Fadlallah, and A. Serhrouchni, "Analysis of cryptographic algorithms on IoT hardware platforms," in *Proc. the 2018 2nd Cyber Security in Networking Conference (CSNet)*, 2018, pp. 1–5.
[13] Q. Yu, X. Wang, and L. Nie, "Optical recording of brain functions based on voltage-sensitive dyes," *Chin Chem Lett*., vol. 32, pp. 1879–1887, 2021.
[14] M. M. Islam *et al*., "FPGA implementation of high-speed area-efficient processor for elliptic curve point multiplication over prime field," *IEEE Access*, vol. 7, pp. 178811–178826, 2019.
[15] A. Kumar, S. Jakhar, and S. Makkar, "Comparative analysis between DES and RSA algorithm's," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 2, no. 7, pp. 386–391, 2021.
[16] D. S. A. Elminaam, H. M. A. Kader, and M. M. M. Hadhoud, "Performance evaluation of symmetric encryption algorithms," *IJCSNS International Journal of Computer Science and Network Security*, vol. 8, no. 12, pp. 280–286, 2004.
[17] C. Pu and X. Zhou, "Suppression attack against multicast protocol in low power and lossy Networks: Analysis and defenses," *Sensors*, vol. 18, 3236, 2018.
[18] S. Pronika and S. Tyagi, "Performance analysis of encryption and decryption algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, pp. 1030–1038, 2021.
[19] P. Arora, A. Singh, and H. Tiyagi, "Evaluation and comparison of security issues on cloud computing environment," *World of Computer Science and Information Technology Journal (WCSIT)*, vol. 2, no. 5, pp. 179–183, 2012.
[20] U. Somani, K. Lakhani, and M. Mundra, "Implementing digital signatures with RSA encryption algorithm to enhance the data security of cloud in cloud computing," in *Proc. 1st International Conference on Parallel, Distributed and Grid Computing (PDGC)*, 2021, pp. 211–216.
[21] L. Yu, Q. Wu, Y. Xu, G. Ding, and L. Jia, "Power control games for multi-user anti-jamming communications," *Wireless Networks*, vol. 25, 2018.
[22] A. Varga. (1997). OMNeT++ Home Page. [Online]. Available: http://www.hit.bme.hu/phd/vargaa/omnetpp.htm
[23] L. S. Brakmo and L. L. Peterson, "TCP vegas: End to end congestion avoidance on a global internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, 2019.
[24] C. Ozgur, M. Kleckner, and Y. Li, "Selection of software for university courses and for Firms that use business analytics," *Sage Open Journal*, pp. 1–12, 2017.
[25] Y. Tian, R. Cheng, X. Y. Zhang, and Y. C. Jin, "PlatEMO: A matlab platform for evolutionary multi-objective optimization," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017.
[26] K. Abdelwahab and G. Ronald, *An Introduction to Numerical Methods: A MATLAB Approach*, Chapman & Hall, 2017.
[27] S. A. Mohammed, and S. B. Sadkhan, "Design of wireless network based on NS2," *Journal of Global Research in Computer Science*, vol. 3, no. 12, pp. 1–8, 2012.
[28] B. Çeliku, R. Prodani, and K. Qafzezi, "Symmetric versus asymmetric cryptographic techniques and security issues under various applications," *Global Journal of Computers and Technology*, vol, 6, no. 1, 2017.
[29] B. Bhagyavati, "Wireless security techniques: An overview," *Wireless Security Technique*, vol. 87, 2015.
[30] S. Swati. "Wireless network security protocols a comparative study," *International Journal of Emerging Technology and Advanced Engineering*, 2012.
[31] Y. Y. Jiang, A. Hu, and J. Huang, "A lightweight physical-layer based security strategy for internet of things," *Cluster Computing*, vol. 22, pp. 12971–12983, 2018.
[32] K. Kumar, N. A. Sharma, and R. Prasad, "A survey on quantum computing with Main focus on the methods of implementation and commercialization gap," in *Proc. 2022 2nd Asia-Pacific World Congress on Computer Science and Engineering*, 2015, pp. 1–7.
[33] M. Mandal and R. Dutta, "Cost-effective private linear key agreement with adaptive CCA security from prime order multilinear maps and tracing traitors," in *Proc. International Conference on Security and Cryptography*, 2018, pp. 356–363.
[34] A. Abdullah, A. Mahalanobis, and V. M. Mallick, "A new method for solving the elliptic curve discrete logarithm problem," *J. Groups Complexity Cryptol*., vol. 12, no. 2, 2021.