# Implementation of AES Encryption Algorithm with Chaos-Based Dynamic Block Key on TLS Protocol

Bayu Samudra
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13520128@std.stei.itb.ac.id

Rinaldi Munir
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
rinaldi@informatika.org

***Abstract*—Nowadays, the Cryptography world is developing very rapidly. One developmentin Cryptography is dynamic keys ciphers. There are some studies about this topic, but so far, there is no protocol that implements this cipher in practice. This paper proposes a dynamic block key cipher that can be used in the TLS protocol. The proposed cipher is implemented in AES encryption algorithm with Sine-Henon map-based CSPRNG as the key generator. Sine-Henon map is used because it can preserve forward secrecy. The proposed cipher also implemented in the TLSv1.2. The evaluation of the proposed scheme is done by doing several experiments and comparing the cipher with the original AES encryption algorithm. The result of the evaluation shows that the proposed scheme has similar quality with the original AES encryption algorithm based on NIST Statistical Test Suite, MAD Analysis, and CCA Analysis. The proposed scheme also can handle replay and tampering attack.**

***Index Terms*—Sine-Henon Map, TLS, AES, Chaos System, Dynamic Block Cipher**

## I. Introduction

Nowadays, Information security is one of the critical aspects in the information system. Information security is needed to protect data from unauthorized access. One of the methods to protect the data is by encryption data. As [6], there are two types of encryption, symmetric and asymmetric encryption. Symmetric encryption is a method to encrypt and decrypt data using the same key. On the other hand, asymmetric encryption is a method to encrypt and decrypt the data with different keys. Based on [4], asymmetric encryption is more secure than symmetric encryption. However, asymmetric encryption is slower than symmetric encryption.

Symmetric encryption has advantages in terms of speed. Nowadays, symmetric encryption is developed rapidly. One encryption scheme that is currently developed is dynamic symmetric encryption. Dynamic encryption is a method to change key dynamically when do encryption. This method may increase the confidentiality of data because this method can hide the statistical pattern of the ciphertext. As [6], this scheme applies the confusion principle from Shannon.

TLS protocol is one of communication protocol that can be use to preserve the confidentiality and the integrity of data. This protocol can be used to transmit data securely from one host to another host. The data that transmitted using TLS protocol is encrypted using symmetric encryption. There is many symmetric encryption algorithm that can be used in TLS protocol, but there is not any implementation that use dynamic encryption in TLS protocol. In this paper, we propose a dynamic encryption scheme that can be used in TLS protocol. We implement the dynamic encryption scheme in AES encryption algorithm. We use chaos-based dynamic block key to change the key dynamically. We evaluate the functionality and robustness of the proposed scheme by doing several experiments and comparing the cipher with original AES encryption algorithm.

## II. Related Works

There are several design of dynamic encryption. As [10], the dynamic encryption scheme that they proposed is by changing the internal structure of AES algorithm in one round. This method changes the S-Box, irreducable polynomial, and affine constant in AES algorithm. The disadvantage of this method is the complexity of the implementation. Besides that, AES encryption cannot be optimized because the internal structure of AES algorithm is changed.

Another dynamic encryption scheme is proposed by [1]. In their paper that they proposed a dynamic encryption scheme by combining AES encryption with OTP Algoritm. The source of OTP is generated using Linear Congruential Generator (LCG). The salt of LCG is appended to the plaintext. After that, the plaintext is encrypted using AES encryption algorithm. The disadvantage of this method is the the security of the scheme is not guaranteed because the source of OTP is generated using LCG. LCG can be predicted by the attacker easily, so the attacker can predict the OTP.

Lastly, dynamic encryption scheme is proposed by [5]. In their paper, they proposed a dynamic encryption scheme by changing the key dynamically. The key is generated based on the chaos-based CSPRNG. The chaos system that they used in this paper is Henon Map. The process of generating the key is by iterating the Henon Map. The result of the Henon Map is converted to integer. This integer is used as the key of AES encryption algorithm. The disadvantage of this method is henon map is not preserving forward secrecy. The attacker can predict the key by knowing two consecutive key. Therefore, they can generate the key and decrypt the ciphertext.

As [5], they proposed a method to sync the chaos system between participants. The method is by transferring some correction parameter to the client. This process needs some RTT to transfer the correction parameter. The disadvantage of this method is the latency of the communication is increased. Besides that, the security of the scheme is not guaranteed because the attacker can predict the key by knowing the correction parameter.

## III. Proposed Cipher

To solve the known problem, we propose the dynamic encryption scheme and the implementation of TLS protocol.

### A. Chaos System

Chaos system that we choose in this paper is developed based on [5] and [8] methods. Instead of using Henon-Map, we use Sine-Henon Map to generate the key. The Sine-Henon Map is defined as Equation 1.

$$\begin{aligned} X_{i+1} &= \text{mod}((1 - a \cdot X_i^2 + Y_i) + (\frac{\mu}{4} \cdot \sin(\pi \cdot X_i)) \cdot 100, 1) \\ Y_{i+1} &= \text{mod}((b \cdot X_i + Z_i) \cdot 100, 1) \\ Z_{i+1} &= \frac{\mu}{4} \cdot \sin(\pi \cdot Z_i) \end{aligned} \tag{1}$$

This equation is built by combining the Henon map and Sine map. This combining method is based on method that proposed by [8]. The combination of this method can preserve the forward secrecy. The reason of this claim is the attacker should have parameter that is unknown to attacker. The parameter is $Z$. When the attacker know the value of $X$, they cannot predict the value of $Z$. Therefore, the attacker cannot predict the next key.

The modulo operation is used to make the value of $X$ and $Y$ in the range of $[0, 1]$. This technique can make us easier to convert the value of $X$, $Y$, $Z$ to integer because we exactly sure the range of generated value.

Based on [7], The multiplier 100 in Equation 1 is used to make the distribution of the generated value uniform. This differences can be seen in the histogram of the generated value. The histogram of the generated value is shown in Figure 1.

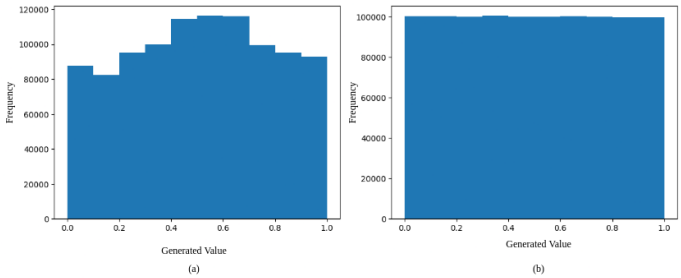


Fig. 1. Histogram of Sine-Henon Map (a) without multiplier 100 and (b) with multiplier 100

The value of parameter that we used in this paper is $a = 1.4$, $b = 0.3$, and $\mu = 3.75$. The reference value of $a$ and $b$ based on experiments by [5]. The value of $\mu$ is choosen based on recommendation by [8].

### B. Dynamic Encryption Scheme

The dynamic encryption scheme that we proposed is by changing the key dynamically. The key is generated based on the chaos-based CSPRNG. The chaos system that we use is Sine-Henon Map. Based on [6], chaos system can be used to generate securely. The method to generate the key is by generating a state of the chaos system. In this case, the state is the value of $X$, $Y$, and $Z$. Next, the value of $X$ is converted to integer. Process of converting this value can be done by using Equation 2.

$$X' = X \cdot 2^N \tag{2}$$

In this case, $N$ is the number of bits that we want to generate. The value of $X'$ is the result of the conversion. This value is used to compose the key. Value of $X$ is the number that is generated by the chaos system. In our implementation, we use 256-bit key. We compose the key by generating 32 bits of $X'$ 8 times. The result of this process is the key that is used to encrypt the plaintext.

The encryption process is done by using AES-256 encryption algorithm. We use counter mode to encrypt the plaintext. Counter mode is choosen because this mode can give us flexibility to encrypt the plaintext in any size. Besides that, to increase the randomness of stream key that is generated by counter mode, we also generate the counter by using Sine-Henon Map. We convert the value of $X$ to integer by using Equation 2 with $N = 16$. Figure 2 shows the process of encryption.

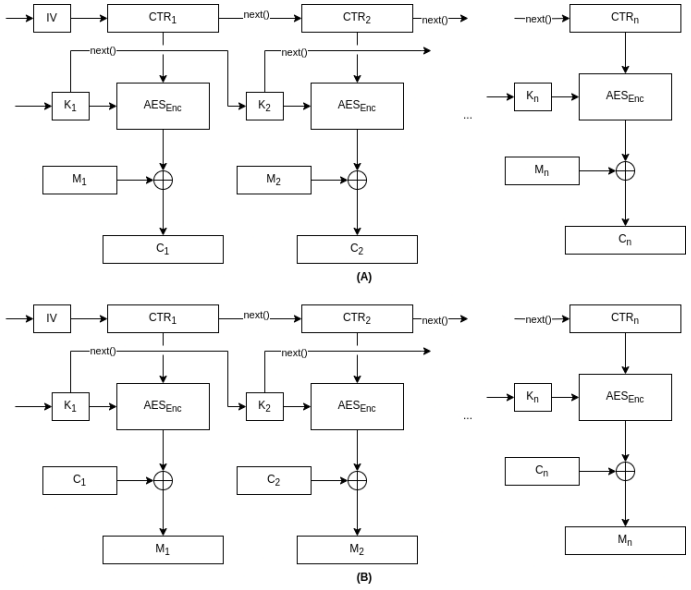


Fig. 2. Encryption Process

Based on that figure, the cipher needs initial value $IV$ as the initial counter and $K_1$ as the initial key. The value of

$IV$ will be a generator of the counter by expanding the value using Sine-Henon Map-based CSPRNG. The value of $K_1$ is used as the initial key. The value of $K_n$ is generated by using Sine-Henon Map-based CSPRNG based on Equation 1. The value of $K_n$ is used as the key of AES encryption algorithm at block $n$.

## IV. Proposed TLS Protocol

In implementation, we follow the TLSv1.2 specification as defined in [3]. Overall, TLS protocol consists of two main protocols, Handshake Protocol and Record Protocol. Handshake Protocol is used to establish the connection between the client and the server. In this protocol, we also establish the chaos system between participiants. Record Protocol is used to transmit the data securely. In this protocol, we use the dynamic encryption scheme that we proposed.

### A. Handshake Protocol

As [3], Handshake protocol is used to establish the connection between the client and the server. We generate keys for the encryption and do the peer verification in this phase. In nutshell, the process of Handshake Protocol is shown in Figure 3.

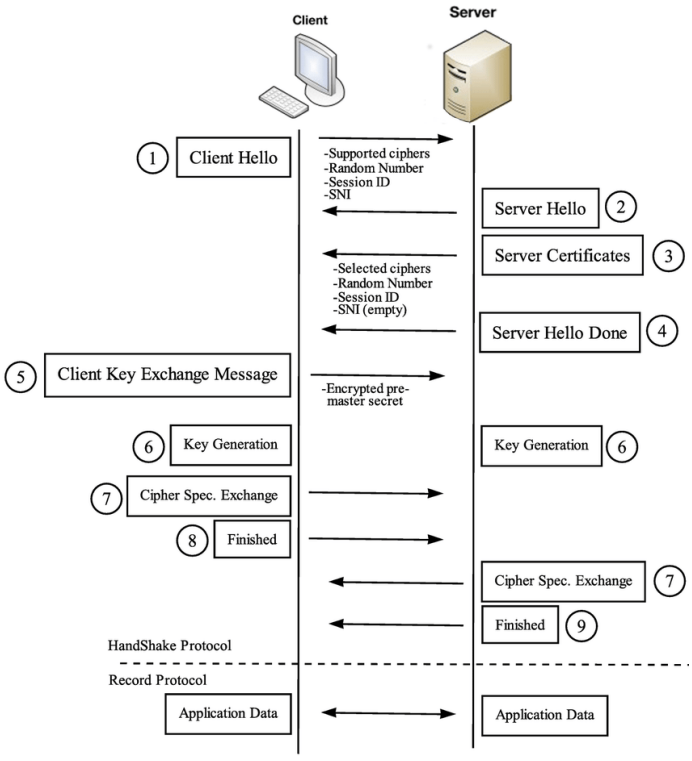


Fig. 3. TLS Protocol (Source [9])

In our implementation, we follow the process of Handshake Protocol as defined in [3]. We use ECDH key exchange to establish the connection. This method is choosen because this method is more secure than RSA key exchange. Result of the ECDH key exchange is premaster key. This key is used to generate the master key. The master key is used to generate the key for encryption and decryption. This process is shown in Figure 4.

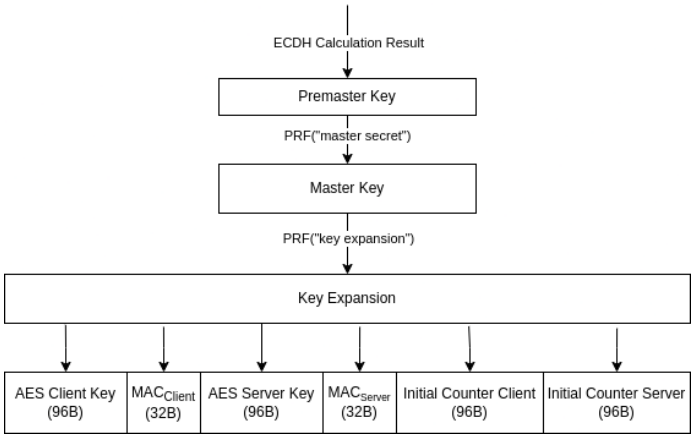


Fig. 4. Key Generation

Based on that figure, we expand master key using PRF function that defined in [3]. The PRF function that we used is HMAC-SHA256. Equation 3 shows the PRF function.

$$\text{PRF}(K, L, S) = \text{P}_{\text{hash}}(K, L||S) \tag{3}$$

In Equation 3, $K$ is the master key. $L$ is the label that we used to generate. The label that we use is shown at Figure 4. $S$ is the seed. The P function is defined as Equation 4.

$$\begin{aligned}\text{P}_{\text{hash}}(K, S) = \text{HMAC}_{\text{hash}}(K, A(1) + S) \parallel \\ \text{HMAC}_{\text{hash}}(K, A(2) + S) \parallel \dots\end{aligned} \tag{4}$$

In Equation 4, $A$ is the function that is used to expand the key. We generate the key until the length that we need. Function $A$ is defined as Equation 5.

$$\begin{aligned}A(0) &= S \\ A(i) &= \text{HMAC}_{\text{hash}}(K, A(i-1))\end{aligned} \tag{5}$$

Based Figure 4, first we should generate premaster key. This premaster key is generated by using ECDH key exchange. After that, we generate the master key by using PRF function. The label that we use is "master secret". The seed that we use is the premaster key. The result of the PRF function is the master key. This master key is used to generate the keys for cipher and HMAC. To generate the key for cipher, we use label "key expansion" and seed based on the master key. The result of the PRF function is the key for cipher and image as shown by Figure 4.

After we expand the master key, we should convert the encryption key to chaos system. The process of converting the key is shown in Figure 5.

To convert the key to chaos system, we devide 96-byte encryption key into 3 parts. Each part is 32-byte. We convert each part to integer by using Equation 2. The result of this process is the value of $X$, $Y$, and $Z$. This value is used as the initial value of the chaos system. To generate initial counter,

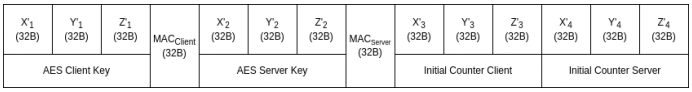


Fig. 5. Converting Key to Chaos System

we also devide 96-byte initial counter into 3 parts. Each part is 32-byte. We convert each part to integer by using Equation 2. The result of this process is the value of $X$, $Y$, and $Z$. This value is used as the initial value of the chaos system to generate the counter. MAC key is not converted to chaos system because it only use to generate the MAC of plaintext.

In this protocol, we generate premaster key by using ECDH key exchange. We use `secp256r1` curve to do ECDH operation. One of the reason that we use this curve is this curve is standardized by NIST. This curve also has small key size, that is 256-bit. Besides that, this curve is supported by most of the library and TLS protocol.

In this protocol, we also verify the server using pinned certificate. The certificate that we use is self-signed certificate. The public key of the certificate is used to verify the signature of the server. The public key is pinned in the client. The client should verify the signature of the server before sending the data. Signature is used to protect ECDH parameter, client hello random, and server hello random. When the signature is not valid, the client should terminate the connection immediately.

### *B. Record Protocol*

Record protocol is used to transimit encrypted data. In this protocol, we use encryption algorithm that we proposed. The frame generation is done by dividing the plaintext into several blocks. Each block is encrypted by proposed encryption algorithm. The process of encryption is shown in Figure 6.

Based on that figure, we save the initial value of the chaos system to generate the key. This initial value should be kept until the frame successfully generated. After that, the plaintext MAC is generated by using HMAC-SHA256. As defined in [3], The formula of MAC is shown in Equation 6.

$$\begin{aligned} H = \text{HMAC}(&\text{key}, \text{frame\_number} \mid\mid \\ &\text{frame.type} \mid\mid \\ &\text{frame.version} \mid\mid \\ &\text{frame.length} \mid\mid \\ &\text{frame.plaintext} \\ &) \end{aligned} \tag{6}$$

In Equation 6, key is the MAC key that we have generated in handshake. frame_number is the number of frame that is currently generated. frame.type is the type of frame. For data transmission, the value of type that used is `33` in decimal. frame.version is the version of the frame. We used value `0x03 0x03` as defined in RFC TLSv1.2. frame.length is the length of the plaintext. frame.plaintext is the plaintext of the frame. The result of the HMAC function is the MAC of the plaintext. This MAC is appended to the plaintext. We do not use compression in this protocol. The reason is for simplicity when analyzing the ciphertext.

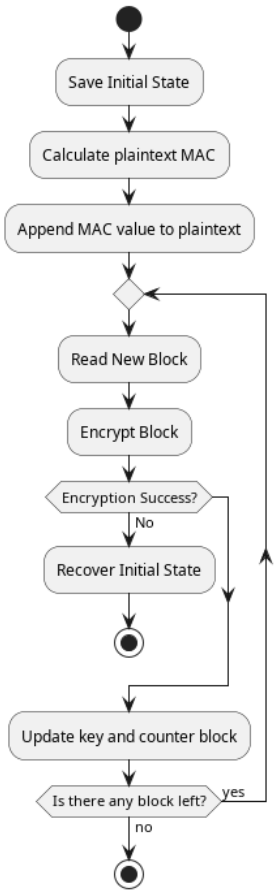


Fig. 6. Record Protocol

After we generate the MAC, we encrypt the plaintext and the MAC using the encryption algorithm that we proposed. The cipher generates new counter and key for each block. The counter is generated by using Sine-Henon Map. We only use parameter $X$ of state to be converted as integer. Conversion is done by using Equation 2. The process continue until the plaintext is fully encrypted. The result of the encryption is the ciphertext.

After we generate the ciphertext, we should compile the frame. The frame is compiled by appending the ciphertext to the frame based on Record Protocol structure. The frame is shown in Figure 7.

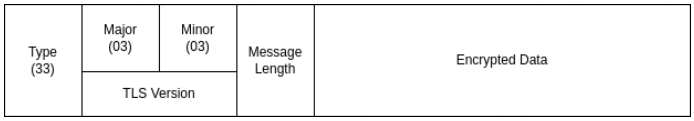


Fig. 7. TLS Frame

In this frame, the first byte is the type of the frame. The type of the frame that we used is `33` in decimal. This value is used to tell that we use TLSv1.2. The second and third byte is the version of the frame. We used value `0x03 0x03` as defined in RFC TLSv1.2. The next blocj is the length of the

frame. The length of the frame is the length of the ciphertext. The rest of the frame is the ciphertext. The frame is sent to the peer.

When peer receive the frame, they should decompile the frame and decrypt the ciphertext. The process of decryption is shown in Figure 8.

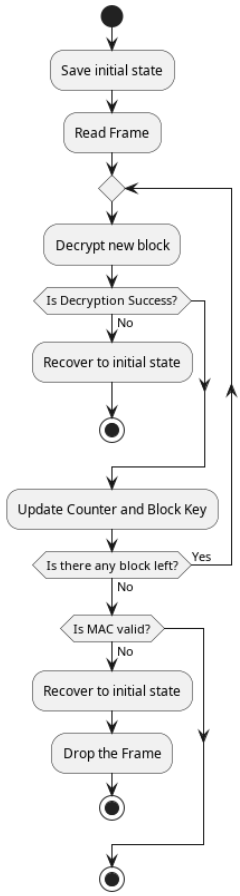


Fig. 8. Record Protocol Decryption

In Figure 8, the peer should save the initial value of the chaos system to generate the key. This initial value should be kept until the frame successfully decrypted. After that, peer should decrypt the ciphertext by using the encryption algorithm that we proposed. The key rotation is done every block. The counter is generated by using Sine-Henon Map. The process continue until the ciphertext is fully decrypted. When the decryption process is failed, the peer should recover the initial value of the counter and the key. After all ciphertext is decrypted, the peer should verify the MAC of the plaintext based on formula at Equation 6. When the MAC is not valid, the peer should recover the initial value of the counter and the key. When all process is success, the data will be passed to the application layer.

## V. Implementation

We implement our proposed scheme in Python language. We build a python library and simple file server to demostrate our scheme. Overall, our implementation is based on component diagram at Figure 9.

This library consists of several components. The first component is the `main` module. This module is used as entry point for simple file server and echo server. This module is used for testing purposes only. In this library, there is important modules, these are `data`, `conn`, and `crypto` module.

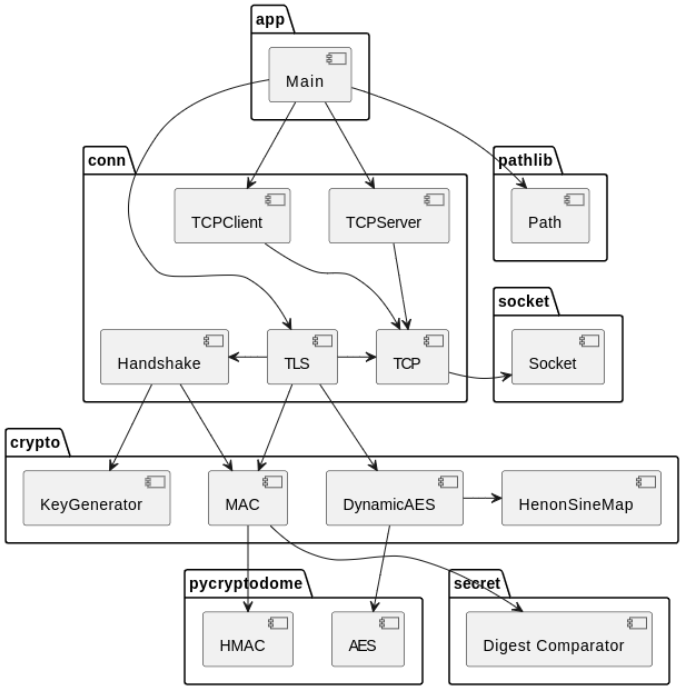


Fig. 9. Component Diagram

Module `data` is responsible for representing data in TLS protocol. This module implements data structure that defined in [3] and [2]. In other hand, module `conn` is responsible to establish the connection between client and server. This module implements TLS protocol and TCP for transport layer of TLS. This module also implement connection using UNIX socket. This feature will be used for scenario testing. This module also responsible for handing handshake and sending encrypted data. Lastly, module `crypto` is the implementation of cipher that we proposed. This module implements the encryption and decryption process of the data. This module also implements the chaos system that we used in the cipher.

## VI. Testing

This section will discuss the testing that we have done to evaluate the functionality and security of the proposed scheme. We evaluate the functionality of the proposed scheme by doing several scenario related to functionality and attack. We evaluate the security of the proposed scheme by comparing the cipher with original AES encryption algorithm. We are comparing the result based on NIST Statistical Test Suite, MAD Analysis, adn CCA Analysis.

### A. Cipher Testing

We evaluate the security of the proposed scheme by comparing the cipher with original AES encryption algorithm. We compare with counter mode in dynamic block and static block cipher. We use NIST Statistical Test Suite, MAD Analysis, and CCA Analysis to evaluate the security of the proposed scheme. Confidence level that is used in this test is 95%. The test is success if the proposed cipher has similar or better quality if we compare with the static block cipher. The result of NIST Statistical Test Suite is shown in Table I.

TABLE I
NIST TEST RESULT

| Test Name | Static Block | Dynamic Block | P-Value |
|---|---|---|---|
| Entropy Test | 98,6% | 98,2% | -0,504049 |
| Block Frequency Test | 99,0% | 98,4% | 0,837512 |
| Cummulative Sum Test | 98,5% | 99,2% | 1.038080 |
| FFT Test | 98,0% | 98,4% | 0,475705 |
| Frequency Test | 98,6% | 99,2% | 0,909550 |
| Linear Complexity Test | 99,0% | 99,2% | 0,334844 |
| Longest Runs Test | 99,2% | 99,0% | -0,334844 |
| Non Overlapping Template Test | 99,0% | 99,0% | -0,037565 |
| Overlapping Template Test | 99,2% | 98,8% | -0,635642 |
| Random Excursions Test | 98,1% | 98,6% | -0,787570 |
| Random Excursions Variant Test | 98,8% | 99,2% | 0,607298 |
| Rank Test | 99,4% | 98,8% | -1.004531 |
| Runs Test | 98,8% | 99,1% | 1.674218 |
| Serial Test | 99,2% | 99,0% | 0,635642 |
| Universal Test | 99,4% | 99,4% | 0,000000 |

Based on Z-test, the critical value for this test is -1.729. Based on the result of the test, the proposed cipher has similar quality with the static block cipher. The p-value of the test is above than the critical value. Therefore, the proposed cipher has similar secure quality based on NIST Statistical Test Suite.

The MAD (Mean Absolute Deviation) Analysis is conducted by comparing the mean of the absolute difference of the proposed cipher and the static block cipher. The testing is conducted by comparing 10 encrypted images using the proposed cipher and the static block cipher. The formula that we used to conduct this test is shown by Equation 7.

$$\begin{aligned} \text{MAD} &= \tfrac{1}{N}\textstyle\sum_{i=0}^{N-1} |X_i - \bar{X}| \\ \bar{X} &= \tfrac{1}{N}\textstyle\sum_{i=0}^{N-1} X_i \end{aligned} \tag{7}$$

The result of the test is shown in Table II. The result of the test is the proposed cipher has similar quality with the static block cipher. The MAD of the proposed cipher is 201.077 and the MAD of the static block cipher is 203.470840. The deviation of the MAD is 59.976003 and 60.427154. Based t-test with 95% confidence level, the critical value is -1.729. The p-value of the test is -0.0889. Thus, the proposed cipher has similar quality with the static block cipher based on MAD Analysis.

TABLE II
MAD TEST RESULT

| Cipher Type | MAD average | MAD deviation |
|---|---|---|
| Static Block | 201.077 | 59.976003 |
| Dynamic Block (Proposed) | 203.470840 | 60.427154 |

TABLE III
MEAN OF RESULT CCA TEST

| Cipher Type | CCA Mean | | |
|---|---|---|---|
| | Horizontal | Vertical | Diagonal |
| Static Block | 0.000165 | 0.000207 | 0.000278 |
| Dynamic Block (Proposed) | 0.000150 | 0.000193 | 0.000310 |

The CCA (Connected Component Analysis) is conducted by comparing the connected component of the proposed cipher and the static block cipher. The testing is conducted by comparing 10 encrypted images using the proposed cipher and the static block cipher. The direction that we used to conduct this test is diagonal, vertical, and horizontal. The formula that we used to conduct this test is shown by Equation 8.

$$\text{CCA} = \frac{\sum_{i=1}^{N}(x_i - \frac{1}{N}\sum_{i=1}^{N} x_i)(y_i - \frac{1}{N}\sum_{i=1}^{N} y_i)}{\sqrt{\sum_{i=1}^{N}(x_i - \frac{1}{N}\sum_{i=1}^{N} x_i)^2 \cdot \sum_{i=1}^{N}(y_i - \frac{1}{N}\sum_{i=1}^{N} y_i)^2}} \tag{8}$$

The test is accepted when the connected component of the proposed cipher is similar or better than the static block cipher. The result of the test is shown in Table III and IV.

The result of the test is the proposed cipher has similar quality with the static block cipher. The connected component of the proposed cipher is 0.0001 and the connected component of the static block cipher is 0.0002. The deviation of the connected component is 0.0001 and 0.0002. Based t-test with 95% confidence level, the critical value is -1.729. The p-value of the test for direction horizontal, vertical, and diagonal respectively are 0.277, 0.218, and -0.324. Thus, the proposed cipher has similar quality with the static block cipher based on CCA Analysis.

### B. Protocol Testing

We evaluate the functionality and security of the proposed TLS protocol by doing several scenario. The scenarios that we implements are handshake scenario, data transmission scenario, and attack scenario. Handshake scenario is used to evaluate the functionality of the handshake protocol. This test will be accepted when the connection is established successfully. Data transmission scenario is used to evaluate the functionality of the record protocol. This test will be accepted

TABLE IV
DEVIATION OF RESULT CCA TEST

| Cipher Type | CCA Deviation | | |
|---|---|---|---|
| | Horizontal | Vertical | Diagonal |
| Static Block | 0.000183 | 0.000251 | 0.000337 |
| Dynamic Block (Proposed) | 0.000199 | 0.000232 | 0.000346 |

when the data is transmitted successfully. Attack scenario is used to evaluate the robustness of the proposed scheme. Attack scenario that we use is tampering attack and replay attack. This test will be accepted when the protocol can handle the attack and still can receive data normally. Result of the testing is shown in Table V.

TABLE V
FUNCTIONALITY TESTING

| Scenario | Result |
|---|---|
| Handshake | Success |
| Data Transmission | Success |
| Tampering Attack | Success |
| Replay Attack | Success |

Based on Table V, the proposed scheme is success to handle the scenario that we have implemented. The handshake protocol is success to establish the connection between client and server. The handsahke protocol generates chaos system successfully based on ECDH parameter. The record protocol is success to transmit the data and receive the data. The protocol is also success to handle the attack. The protocol can detect the tampering attack and replay attack. The protocol can handle the attack and still can receive data normally.

## VII. CONCLUSION

Dynamic block encryption can be implemented in TLS protocol. This can be done by using Chaos-based AES encryption algorithm. Chaos system that proposed in this paper is Sine-Henon Map. The key is generated by using the state of the chaos system. The state is used as the initial value of the chaos system. The key is generated by converting the value of the state to integer. The encryption process is done by using AES encryption algorithm. The counter is generated by using Sine-Henon Map. The counter is used as the counter of the counter mode. The result of the encryption is the ciphertext. The decryption process is done by using the same process as the encryption process. The result of the decryption is the plaintext.

Based on the testing that we have done, the proposed scheme is success to handle the scenario that we have implemented. The proposed cipher has similar quality with the static block cipher based on NIST Statistical Test Suite, MAD Analysis, and CCA Analysis. The proposed TLS protocol is success to establish the connection between client and server. The protocol is also success to transmit the data and receive the data. The TLS protocol can handle tampering attack and replay attack. The protocol can detect the attack and still can receive data normally.

## VIII. CODE REPOSITORY

This implementation of this paper can be accessed at https://github.com/bayusamudra5502/dynamic-encryption-protocol

## REFERENCES


[1] M. M. Bachtiar *et al.*, "Security enhancement of aes based encryption using dynamic salt algorithm," *International Conference on Applied Engineering*, 2018.

[2] S. Blake-Wilson *et al.*, "Elliptic curve cryptography (ecc) cipher suites for transport layer security (tls)," Internet Requests for Comments, IETF, RFC 4492, 2006. [Online]. Available: https://datatracker.ietf.org/doc/rfc4492

[3] T. Dierks and E. Rescorla, "The transport layer security (tls) protocol version 1.2," Internet Requests for Comments, IETF, RFC 5246, 2008. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5246

[4] B. Halak *et al.*, "Comparative analysis of energy costs of asymmetric vs symmetric encryption-based security applications," *IEEE Access*, vol. 10, pp. 76 707–76 719, 2022.

[5] C.-H. Lin, G.-S. Hu, C.-Y. Chan, and Y. Jun-Juh, "Chaos-based synchronized dynamic keys and their application to image encryption with an improved aes algorithm," *Appl. Sci*, vol. 11, no. 3, 2021.

[6] R. Munir, *Kriptografi*, ser. Volume 2. Penerbit Informatika, 2019.

[7] P. Nurhaliza, "Implementasi pembangkit bilangan acak semu dengan henon-sine hyperchaotic map," *Makalah Tugas IF4020 Kriptografi*, 2023.

[8] S. Patel *et al.*, "Colour image encryption based on customized neural network and dna encoding," *Neural Computing and Applications*, vol. 33, 2021.

[9] W. M. Shbair *et al.*, "A multi-level framework to identify https services," *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2016.

[10] A. Singh *et al.*, "Image encryption and analysis using dynamic aes," *2019 5th International Conference on Optimization and Applications (ICOA)*, pp. 1–6, 2019.