CEDTECH International Journal of Engineering & Applied Science
Volume 4, Number 2, June 2023
http://www.cedtechjournals.org
ISSN: 2756-4533

A DYNAMIC RANDOM ADVANCED ENCRYPTION STANDARD (DR-AES) -A NEW SOLUTION

***Adamu, M., Kolo[1], H., Isah[2], A.D., Gonna, I.S.**
*Department of Computer Science, Federal Polytechnic, Bida, Niger State
[1]Department of Computer Science, Federal Polytechnic, Bida, Niger State
[2]Department of Computer Science, Niger State Polytechnic, Zungeru, Niger State
[3]Department of Computer Science, Niger State Polytechnic, Zungeru, Niger State
Email: bejian2004@gmail.com

ABSTRACT

The Advance Encryption Standard (AES) is a widely used encryption algorithm that provides strong protection for sensitive data. The AES is the most popular symmetric cryptographic scheme that uses the same key for encryption and decryption process. However, the conventional AES suffers from key distribution and resources consumptions problem. One of the key performance issues of the traditional AES algorithm is its complexity and speed of execution. These issues made AES unsuitable for real-time applications and applications that required less complexity. This paper proposes anew enhance AES encryption scheme called Dynamic Random Advance Encryption Standard (DR-AES) and implement in Matlab environment. The DR-AES introduces randomness into the encryption process, making it more difficult for attackers to launch certain types of attacks and also reduced number of rounds to reduce the complexity of AES and make it suitable for less complex and real-time applications. The performance of the DR-AES is far better in terms of encryption and decryption operation when compared with the conventional AES. From a security point of view, the proposed DR-AES complies with Kerckhoff's principle. This means the scheme has open design and the security provided by the DR-AES depends only on the secrecy of the encryption key.

Keywords: *Advanced Encryption Standard, Cryptography, Dynamic Encryption, Cipher Text, Plain Text, Encryption, Decryption.*

INTRODUCTION

Humans have an innate desire to communicate with one another as a way of understanding each other. With the advancements in technology, such as telecommunications and computers, people can now store data digitally. However, this digital data storage also comes with risks, particularly when it comes to the security of internet communication. (Muttaqin et al., 2020). Internet communication plays a significant role in transferring large amounts of data across various industries. However, some of this data may be sent through insecure channels, making it vulnerable to intruders. To protect sensitive information, both private and public sectors use various techniques and methods. One of the most important and widely used techniques is cryptography, which involves the use of encryption and decryption to secure data from attackers (Abdullah, 2017).

Encryption is a method used to ensure the safety of sensitive information by using algorithms to perform byte substitutions and matrix transformations on the original message, turning it into a jumbled, unreadable message known as ciphertext. Information security can be maintained through the use of commonly available encryption algorithms, but the choice of key is crucial as it directly determines the security of the encryption (D'souza, 2017). Some of the most widely used symmetric key encryption algorithms are Data Encryption Standard (DES), Triple DES, and Advance Encryption Standard (AES). DES is a classic example of a block cipher, which was the result of a competition set by the US National Bureau of Standards in 1973. It was widely adopted in 1977 (Abomhara et al., 2010). AES is a widely used encryption method in recent times. It is made up of various bit substitutions, permutations, round key functions, and transformations. AES is a faster algorithm and has not yet been broken by any known practical attacks. Therefore, it is still considered one of the most reliable encryption standards for advanced information security (Nivedhaa & Justus, 2018).

The AES algorithm has multiple options for configuration. The first option is the key size or key length, which offers three choices: 128, 192, and 256 bits. The second option is the chaining mode, which determines how multiple block ciphers operate in a chained mode to encrypt information larger than 16 bytes. Some common chaining modes include: Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), and Output Feedback (OFB) (García, 2015).
Several modifications have been carried out on AES algorithm, most of which were done to improve the AES implementation in terms of memory reduction, computational power minimization, encryption and decryption time reduction and security. However, most of this research still uses maximum AES rounds which is still complex for light weight or

resource starved applications. Hence, this paper proposes a new Dynamic Random AES to reduce the number of rounds required for encryption and decryption while improving the security for mobile devices.
The remainder of this paper is structured as follows: Section 2 discuss the existing modifications of AES and the traditional AES, section 3 discussed the proposed DR-AES scheme. In section 4, the expected results are discussed. This is followed by conclusion in section 5.

Literature Review

Several modifications to enhance AES encryption scheme has been proposed. Abikoye et al. (2019) proposed an enhanced version of AES by modifying the SubBytes and ShiftRows transformations. The SubBytes transformation was made round key dependent, while the ShiftRows transformation was randomized. The goal of these modifications was to make the two transformations dependent on the key, so that a small change in the key would result in a significant change in the ciphertext. In this approach, the AES SubBytes transformation was made round key dependent, to make it easier to detect changes in the key. To achieve this, the 16 bytes round key was used to generate four eight-bit keys. The modified algorithm was tested for both its avalanche effect and how long it takes to run, and it was found that while the avalanche effect was improved, the execution time was slightly longer.

Talirongan et al., (2019) proposed a modification to the round function of the AES algorithm to enhance the security of the encryption and decryption processes by utilizing the Butterfly Effect. This method improves the security level by increasing the degree of diffusion, the degree of confusion and integrity check. The modification uses the butterfly effect to improve the key space and nonlinear system of the AES algorithm. The Lorenz attractor, also known as the butterfly effect, was applied to achieve the enhancement. The proposed method begins by taking in inputs such as plain text and a key, which is then utilized in the butterfly effect. Each round has unique values generated by the butterfly effect, as the input key goes through the butterfly effect function. Three security measures (degree of diffusion, degree of confusion and integrity check) are used to evaluate the security level of the modified algorithm. The results of the testing demonstrate that the modified AES algorithm is more secure compared with AES algorithm.

Ali & Shaker, (2020) proposed an improvement for the Advanced Encryption Standard (AES) algorithm which utilizes an additional key generated by a linear feedback shift register (LFSR). This approach allows for more efficient random number generation and reduces the number of rounds needed. The LFSR is used to generate 256 random keys which are added to both the original text and the original AES key (key 1). The use of LFSR to generate an additional key randomly and incorporating it into each round at the "add round key" step makes the algorithm more secure. Additionally, reducing the number of rounds makes the modified algorithm faster than the original, making it more suitable for protecting data in distributed systems.

Indrasena Reddy & Siva Kumar (2020) presented a new method for creating keys using a combination of the Advanced Encryption Standard (AES) and the Flower Pollination Algorithm (FPA), which they referred to as the Modified AES (MAES). The key generation process in the MAES algorithm is crucial as it affects the creation of S-boxes. By creating key-dependent S-boxes, the strength of the algorithm is increased. The key, ki, is generated using a Pseudo Random Number Generator (PRNG), which uses mathematical calculations to generate a sequence of numbers that mimic randomness. The FPA is used in the MAES key generation process to maximize entropy, and the best key value is obtained by maximizing randomness. The authors also proposed that this approach could be applied to other areas.

Altigani et al. (2021), introduced a new variant of the Advanced Encryption Standard (AES) called the polymorphic AES (P-AES). The P-AES changes the values of AES parameters with each new key, making the exact values known only to authorized parties during runtime. The P-AES supports key lengths of 16, 24, or 32 bytes, but for the purpose of consistency, it is assumed that the key length is 32 bytes. In the operation of P-AES, it Modified MixColumns matrix rearranges the rows of the traditional AES MixColumns matrix, with row0 becoming row1, row1 becoming row2, and row2 becoming row3. Additionally, the P-AES cipher includes a layer of obscurity to prevent attackers from easily determining how it operates. Similar modifications could be proposed to improve performance and security while keeping implementation costs low.

Zinabu and Asferaw (2022) suggested an improvement to the Advanced Encryption Standard (AES) algorithm called Enhanced Efficiency of AES (EE-AES) by replacing the mix column stage of the original AES design with a bitwise reverse transposition technique. This modification reduces the computational requirements of the mix column stage while maintaining the same level of security as AES. The proposed EE-AES algorithm utilizes the bitwise reversed transposition operation. The method utilizes hexadecimal notation (00-09 and 0A-0F) to map two hex digits to eight binary bits, rather than writing out the binary digits. This improves the speed and efficiency of the existing Advance Encryption Standard (AES) and Modified Advance Encryption Standard (MAES) algorithms. As a next step, one could test the algorithm with different bit sizes and compare it to other state-of-the-art algorithms.

Traditional AES

AES is a widely used symmetric encryption algorithm that uses a fixed-size key to encrypt and decrypt data. It is a block cipher, which means that it operates on fixed-size blocks of data, using a secret key to transform the data in a way that is designed to be difficult to reverse without knowledge of the key. It processes data in blocks of 128 bits, with the option of using key lengths of 128, 192, or 256 bits. The algorithm operates on a 2-D array of 4x4 bytes, known as the State. The State starts as plaintext and is transformed into cipher text through the encryption process. Each row of the State contains 4 bytes, and the four bytes in each column form a 32-bit word (Forhad et al., 2018). Figure 1 shows the traditional AES encryption scheme.

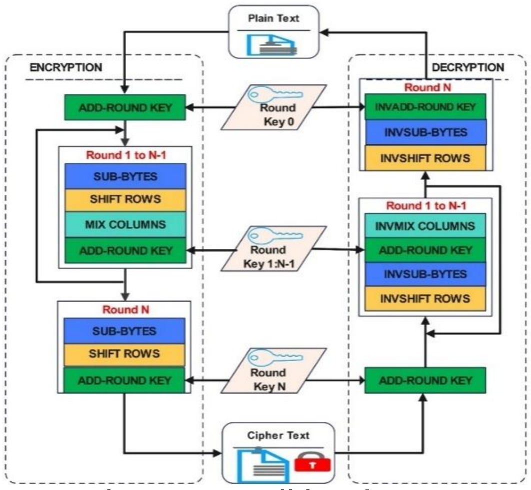

Figure 1: Traditional AES

The encryption process begins with an initial round key addition, followed by a round function that is applied to the State. The round function consists of four transformations: SubBytes, Shift Rows, MixColumns, and Add RoundKey. The number of times the round function is applied depends on the key length used, with AES-128 applying it 10 times, AES-19 2 applying it 12 times, and AES-256 applying it 14 times. The transformations used in the round function are reversible linear and non-linear operations, which allows for decryption using their inverses (Forhad et al., 2018).

Dynamic Random Advance Encryption Standard (DR-AES)

One of the key performance issues of the traditional AES algorithm is its complexity and speed of execution. These issues made AES unsuitable for real-time applications and applications that required less complexity. To address these issues, a Dynamic Random AES (DR-AES) encryption scheme is proposed for encryption and decryption. In this process, the initial secrete key is expanded using the standard key expansion algorithm. The 128-bits key is expanded to produce 11 new 128 bits (16-bytes) which corresponds with the number of rounds plus one for the preliminary round. For each of the number of chosen rounds (nR), the appropriate round key corresponding to the round number will be selected for the add round operation to change the state. Each 16 byte of the expanded key will be indexed for reference. Figure 2 shows the proposed DR-AES scheme for encryption and decryption.

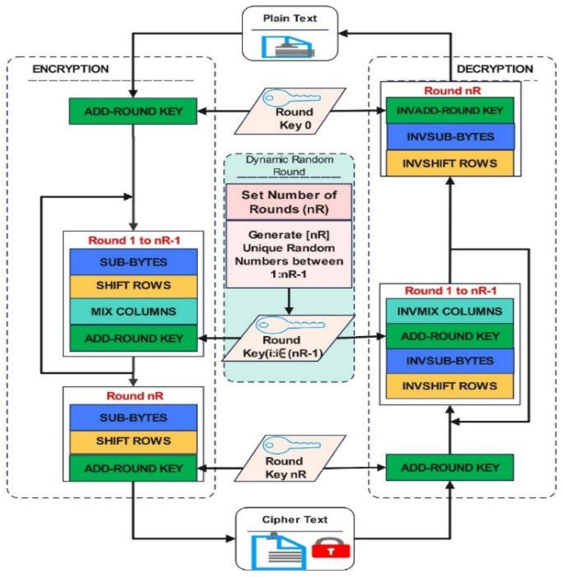

Figure 2: Randomized Dynamic AES Scheme

The proposed encryption scheme comprises of four steps:

Key Generation and Expansion Step

The key expansion used in this research work was adopted from (Yan & Chen, 2016) with some modification. The initial 128-bit key is divided into 4 words and then further expanded into 44 words. Each word is arranged into a 4x4 matrix, with 4 words representing a single key. A total of 11 subkeys are needed, and each key is expressed in 16 bytes. The first key is represented as W_0, W_1, W_2, and W_3. An index number r_0 to r_{10} was assigned to each word as shown in Figure 3

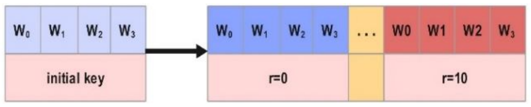

Figure 3: Key expansion

This step is used to generate a series of round keys from the initial encryption key. Mathematically, the key expansion can be expressed as follows: Given the initial key (k_0), which is generated as 4 words secrete key by the user given as:

$K1 = W_0W_1W_2W_3$ (1)

The round constant (Rcon) is generated by XORing the first word (W_0) and last word (W_3) as shown in Equation 3.2. This is followed by shifting the four bytes one position to the left cyclically (nibble bit rotation), followed by substituting each byte from S-box

$Rcon_1 = W_0 \otimes W_3$ (2)

$G_1 = Rcon_1 \otimes SubNib(RotNibW_3)$ (3)

$W_4 = W_0 \otimes G_1$ (4)

$W_5 = W_1 \otimes W_4$ (5)

$W_6 = W_2 \otimes W_5$ (6)

$W_7 = W_3 \otimes W_6$ (7)

$K_1 = W_4W_5W_6W_7$ (8)

The remaining keys

($K2, K3, K4, K5, K6, K7, K8, K9, K10$) are generated in a similar manner with the next 4 words representing key2 ($K2 = W_8W_9W_{10}W_{11}$), key3 ($K3 = W_{12}W_{13}W_{14}W_{15}$), and so on.

Add-Round Key

Add round key: Add Round Key transformation is an XOR operation that adds the round key to the State in each round. In this step, the round key, which is derived from the original encryption key, is added to the state using a bitwise XOR operation. This step is used to introduce confusion into the encryption process as shown in figure 4. Mathematically, this can be represented as:

$X_I = T \otimes K_0$
(9)

where X is the resultant state array (ciphertext), T is the input plaintext, K_0 is the round key at index 1 of the first 4 words.

$$\begin{array}{|c|c|c|c|}\hline T0 & T4 & T8 & T12 \\ \hline T1 & T5 & T9 & T13 \\ \hline T2 & T6 & T10 & T14 \\ \hline T3 & T7 & T11 & T15 \\ \hline\end{array} \otimes \begin{array}{|c|c|c|c|}\hline K_0 & K_4 & K_8 & K_{12} \\ \hline K_1 & K_5 & K_9 & K_{13} \\ \hline K_2 & K_6 & K_{10} & K_{14} \\ \hline K_3 & K_7 & K_{11} & K_{15} \\ \hline\end{array} = \begin{array}{|c|c|c|c|}\hline X_0 & X_4 & X_8 & X_{12} \\ \hline X_1 & X_5 & X_9 & X_{13} \\ \hline X_2 & X_6 & X_{10} & X_{14} \\ \hline X_3 & X_7 & X_{11} & X_{15} \\ \hline\end{array}$$

Figure 4: Initial Add-round Operation

Dynamic Random Round Generation

The dynamic random round generation stage comprises of two steps:

i. Determination of the number of rounds (nR)

In this step, the number of rounds to run is determined. For a 128-bits AES key size, the normal round is 10. In this work, the appropriate number of rounds will be experimented to determine the effect of each lower round in terms of encryption and decryption time and in terms of key size and security effect.

ii. Random round Generation and Key Selection

Generation of nR unique random numbers between one and the total number of rounds corresponding to the key length. Figure 3.3 shows the operation of the random round generation and key selection. Given the total number of rounds N=10, and the number of round = nR, the random round index is expressed as:
For nR = 5 can be given as:

$D_i = [D_1 \quad D_2 D_3 \ldots D_{nR}]$, $where\ i = 1{:}nR$ (10)

$D_5 = [2 \quad 3 \quad 7 \quad 8 \quad 9]$ (1

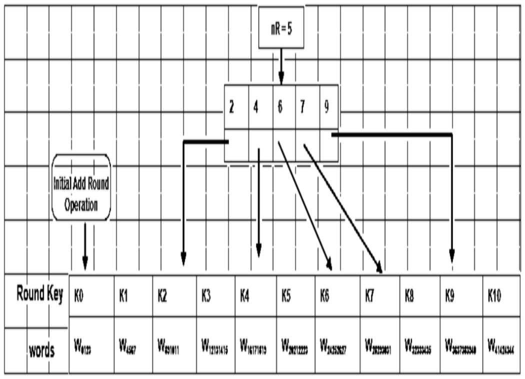

Figure 5: Round key selection

Round Function

The round functions are sub-bytes, shift rows, mixcolumns and add round keys.

i. **Sub-bytes**

The Sub-bytes transformation is a nonlinear byte substitution that operates on each byte of the State using a table, known as an S-box. The S-box is computed using a finite field inversion and an affine transformation. In this step, each byte in the state is replaced with a corresponding value from the S-box. The result is in a matrix of four rows and four columns. To achieve this, first, the finite field inversion is calculated using Equation (12): $I(x) = x^{-1} \ mod\ 2^k$ (12)

ii. **Shift rows**

The Shift Rows transformation is a circular shifting operation that rotates the rows of the State with different numbers of bytes (offsets). In this step, the rows of the state are shifted by a certain number of positions. The number of positions shifted depends on the row number, with the first row being un-shifted, the second row being shifted one position, the third row being shifted two positions, and so on

iii. **MixColumns**

In mixcolumns, each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in thelast round. This step is used to mix the columns of the transformed data before it is encrypted.

iv. Add round key

Add Round Key transformation is an XOR operation that adds the round key to the State in each round. In this step, the round key, which is derived from the original encryption key, is added to the state using a bitwise XOR operation. This step is used to introduce confusion into the encryption process.

Mathematically, this can be represented as:

$$C_I = X_I \otimes K_0, (i: i = 0: N - 1) \quad (13)$$

where C is the resultant state array (ciphertext), X is the input state array (plaintext), K_0 is the round key, and N is the number of columns (32-bit words) in the state array.

DR-AES Encryption Algorithm

ALGORITHM: RAES ENCRYPTION ALGORITHM
INPUT: PLAINTEXT, KEY
OUTPUT: CIPHER TEXT

```
1. Define plaintext
2. Define key
3. Call the function tic to start a timer to measure the execution time.
4. Convert key to hexadecimal and assign the result to variable KEY.
5. Convert plaintext to hexadecimal and assign the result to variable
   PLAINTEXT.
6. Calculate the number of words in the key by dividing the length of
   KEY by 8 and assigning the result to variable Nk.
7. Call the function Key Expansion with arguments KEY and Nk to
   expand the key and assign the result to variable w.
     a. Convert the key from hexadecimal to decimal using the
        hex2dec function and reshape it into a 2D array with 2
        columns.
     b. Reshape the key from step a into a 4xNk matrix.
     c. For i from Nk to 4*(Nk+7)-1 do the following:
```

i. Set the temporary variable temp to the ith column of w.
ii. If i is a multiple of Nk then:
1. Shift the ith column of w circularly to the right by 1 and apply the SubBytes function to it.
2. Calculate the polynomial n = xtime(2, n) where n is the last byte of temp and xtime is a function that multiplies its input by 2 in the Galois field GF(2^8). 3. Bitwise XOR the last byte of temp with n and set the result to the last byte of temp.
iii. Else if Nk > 6 and i is a multiple of 8 then apply the SubBytes function to temp.
iv. Bitwise XOR the ith column of w with the (i-Nk)th column of w and temp and set the result to the (i+1)th column of w.

d. Return w.

8. Randomly select 3 values from the range 1 to 10 and assign them to variable DR.
9. Call the function tic to start a timer to measure the execution time.
10. Call the function newCipher with arguments PLAINTEXT, R, and w to encrypt the plaintext.
    a. Convert In from hexadecimal to decimal and reshape the result to a matrix with 4 rows and a number of columns.
    b. Assign the result to variable state.
    c. Call the function AddRoundKey with arguments state and the first 4 columns of w to add the key to the state.
        1. Define the S-box matrix using hexadecimal values
        2. Reshape the S-box matrix by transposing it and then concatenating the rows
        3. Convert the concatenated matrix from hexadecimal to decimal using hex2dec function
        4. Reshape the decimal matrix into a 16x16 matrix
        5. Define the input state matrix
        6. Substitute each element of the state matrix with the corresponding value from the S-box matrix using the formula state=Sbox(state+1)
        7. Perform row shifts on the state matrix using the function ShiftRows(state)
        8. Perform column mixing on the state matrix using the function MixColumns(state)

d. Loop from k=1 to NR-1, where NR is the number of values in R.
e. Call the function SubBytes with argument state to substitute bytes in the state.
f. Call the function ShiftRows with argument state to shift the rows in the state.
g. Call the function MixColumns with argument state to mix the columns in the state.
h. Call the function AddRoundKey with arguments state and the columns of w specified by DR(k) and DR(k)+1 to add the round key to the state.
i. End loop.
j. Call the function SubBytes with argument state to substitute bytes in the state.
k. Call the function ShiftRows with argument state to shift the rows in the state.
l. Call the function AddRoundKey with arguments state and the columns of w specified by R(end) and R(end)+1 to add the round key to the state.
m. Reshape state to a column vector and assign the result to variable Out1.
n. Convert Out1 to lowercase hexadecimal.
o. Convert Out1 from a column vector to a row vector.

11. Assign the result to variable Out1 (**CIPHERTEXT).**
12. Call the function toc to stop the timer and calculate the encryption time. Assign the result to variable encrypt Time.

DR-AES Decryption Algorithm

ALGORITHM: DRAES DECRYPTION ALGORITHM
INPUT: CIPHERTEXT, DR, w
OUTPUT: PLAIN TEXT

1. Start by defining a function named **newInvCipher** that takes in two inputs, **CIPHERTEXT** and **DR**, along with a key expansion **w**.
2. Determine the size of the **DR** matrix and store the value in the variable **NR**.
3. Convert the input **CIPHERTEXT** from a hexadecimal string to a decimal array using the **hex2dec** function and reshape it into a 4x4 matrix. Store the result in the variable **state**.

4. Perform an initial **AddRoundKey** operation by selecting the key schedule **w** corresponding to the last round (**DR(end)**) and XOR it with the **state**.
5. Iterate through the remaining rounds in reverse order from **NR-1** down to **1**.
6. Perform an **InvShiftRows** operation on the **state**.
7. Perform an **InvSubBytes** operation on the **state** using the inverse S-Box lookup table defined in the **Sbox** function.
8. Perform an **InvMixColumns** operation on the **state**.
9. Repeat steps 6-9 for all rounds except the first.
10. Convert the **state** matrix back into a 1x16 decimal array and convert it to a lowercase hexadecimal string using the **dec2hex** and **lower** functions. Store the result in the variable **PLAIN TEXT**.
11. Return the **PLAIN TEXT**string as the output of the **newInvCipher** function.
12. Call the function toc to stop the timer and calculate the decryption time. Assign the result to variable decrypt Time.

Experimental Results and Analysis

The algorithm has been implemented using Matlab environment.

4.1 Time Execution Performance

Performance is a critical factor that can determine the success of any algorithm.

Table1 shown the tine needed to encrypt and decrypt the plaintext of the proposed Dynamic Random Advanced Encryption Standard (DR-AES) with that of AES

Algorithm	Round Number (RN)	Encryption Time	Decryption Time
DR-AES	1	0.0074	0.0161
	2	0.0133	0.0180
	3	0.0167	0.0218
	4	0.0198	0.0257
	5	0.0222	0.0276
	6	0.0246	0.0325
	7	0.0268	0.0382
	8	0.0291	0.0418
	9	0.0305	0.0540
AES	10	0.0588	0.0726

Table 1: Execution Time

As can be seen, the time required to encrypt plaintext of the same size for the proposed DR-AES for NR=1 to 9 is reasonable less. Also, the time needed to decrypt the cipher text is less.

Theorem: Experimental result of the time execution performance for the proposed DR-AES prove that the encryption and decryption process is directly proportional to the round number of the cipher which implies the less the number of round the less time it takes encrypt and decrypt plaintext.

Performance Comparing of the DR-AES versus AES

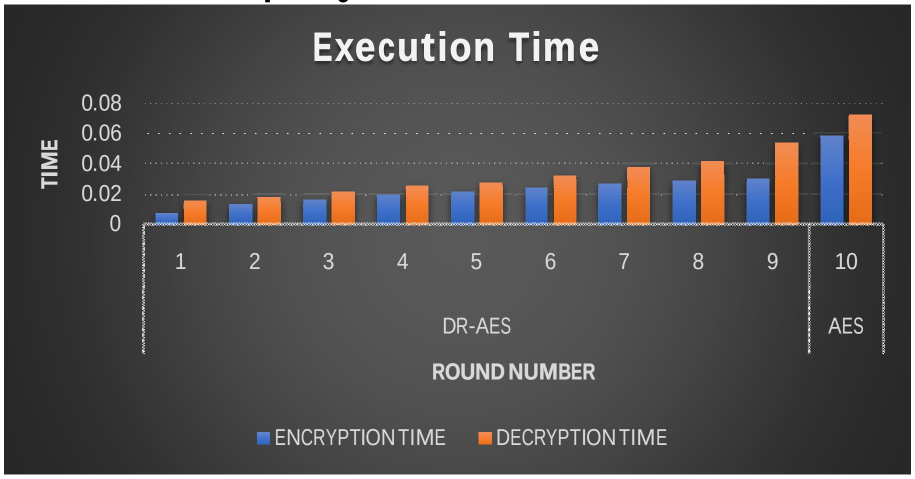

Figure 6: Comparing Performance of the DR-AES and AES

Based on figure 6, the performance of the proposed DR-AES in terms of encryption and decryption operation is far-far less when compared to the traditional AES. This credit (fastness) is attributed to reduction in the number of the round of the DR-AES. The high speed of the scheme makes it suitable for resource constraints applications and devices that is the applications and devices subjected power energy (battery), computational power (processor) and memory space

CONCLUSION

The paper developed a dynamic randomise Advanced Encryption Standard, namely DR-AES. The study aims to changes the static features of the traditional AES and replaced it with dynamic randomise cipher that inherits the AES strengths. As proven, the encryption and decryption time is much better when compared the conventional AES which makes it more suitable for devices with subjected resources constraints and this achievement is as result of reduction in the number of rounds required. The dynamic nature of the scheme makes it very resistance different form of attacks. In future work, Avalanche performance metric can be experiment to investigate the security performance without increasing the implementation cost. Also, suggested to be test in the mobile applications.

REFERENCE

Abdullah, A. M. (2017). *Advanced Encryption Standard (AES) Algorithm to Encrypt and Decrypt Data*.

Abikoye, O. C., Haruna, A. D., Abubakar, A., Akande, N. O., & Asani, E. O. (2019). *SS symmetry for Information Security*. 1–16.

Abomhara, M., Zakaria, O., & Khalifa, O. O. (2010). *An Overview of Video Encryption Techniques*. *2*(1), 103–110.

Ali, H. H., & Shaker, S. H. (2020). Modified Advanced Encryption Standard algorithm for fast transmitted data protection. *IOP Conference Series: Materials Science and Engineering*, *928*(3). https://doi.org/10.1088/1757-899X/928/3/032011

Altigani, A., Hasan, S., Barry, B., Naserelden, S., Elsadig, M. A., & Elshoush, H. T. (2021). A Polymorphic Advanced Encryption Standard - A Novel Approach. *IEEE Access*, *9*, 20191–20207. https://doi.org/10.1109/ACCESS.2021.3051556

D'souza, F. J., & Panchal, D. (2017). *Advanced Encryption Standard (AES) Security Enhancement using Hybrid Approach*. 647–652.

García, D. F. (2015). *Standard Algorithm*. 247–252. https://doi.org/10.1109/MCSI.2015.61

Indrasena Reddy, M. (2020). a Modified Advanced Encryption Standard Algorithm. *Journal of Mechanics of Continua and Mathematical Sciences*, *spl5*(1), 112–117. https://doi.org/10.26782/jmcms.spl.5/2020.01.00027

Muttaqin, K., Rahmadoni, J., Samudra, U., & Andalas, U. (2020). *Analysis And Design of File Security System Aes (Advanced Encryption Standard) Cryptography Based*. *1*(2), 114–123.

Nivedhaa, R., & Justus, J. J. (2018). A Secure Erasure Cloud Storage system using Advanced Encryption Standard algorithm and Proxy Re-encryption. *2018 International Conference on Communication* *and Signal Processing (ICCSP)*, 755–759.

Talirongan, H., Sison, A. M., & Medina, R. P. (2019). Modified advanced encryption standard using butterfly effect. *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM 2018*. https://doi.org/10.1109/HNICEM.2018.8666368

Zinabu, N. G., & Asferaw, S. (2022). *Enhanced Efficiency of Advanced Encryption Standard (EE-AES) Algorithm*. *7*(mix), 59–65. https://doi.org/10.11648/j.ajetm.20220703.13