International Journal of Computer Applications Technology and Research
Volume 14–Issue 03, 132 – 139, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1403.1012

Development of Enhanced Advanced Encryption Standard (AES) Cryptographic Model for Smartphone

C.V Mbamala
Department of Information Technology,
Federal University of Technology Owerri
Imo state. Nigeria

M.O Onyesolu
Department of Computer Science
Nnamdi Azukiwe University, Awka
Anambra, Nigeria

G.N Ezeh
Department of Information Technology,
Federal University of Technology Owerri
Imo state, Nigeria

U.V Maduabuchi
Department of Computer Education
Federal College of Education (Technical) Umunze
Anambra, Nigeria

C.D Anyiam
Department of Computer Science
Federal University of Technology,
Owerri, Nigeria

Abstract: Smartphones typically store data locally or transmit it to the cloud, both of which pose significant security and privacy risks. Data transmission to cloud storage is vulnerable to interception, while local storage is susceptible to physical theft. To mitigate these risks, encryption of data at rest and in transit is essential. The Advanced Encryption Standard (AES) is widely employed for such purposes; however, its computational demands can strain the limited resources of smartphones. This study introduces an optimized version of AES, referred to as Enhanced AES, (E-AES) which incorporates modifications to the shift row transformation, reduces the number of encryption rounds, and integrates an XOR operation. These enhancements aim to improve both security and computational efficiency. Both the conventional AES and enhanced were implemented and compared based on several performance metrics, including avalanche effect, encryption and decryption time, and memory usage. The enhanced AES demonstrated an avalanche effect of 56.25%, higher than the 49.22% achieved by conventional AES, indicating improved security. Additionally, enhanced AES showed a slight speed advantage in encryption and decryption time. In terms of memory usage, E-AES used an average space of 1862.57(MB) when encrypting 1000KB while Conventional AES based application used an average space of 1864.34(MB). This means in every 1000KB encrypted our application saves 1.77(MB) of memory space.

Keywords: Smartphone, Cryptography, Security, Encryption. Decryption.

1. INTRODUCTION

Smartphones have significantly transformed modern life, especially in areas like business, communication, social networking, and access to information. However, with the vast amount of sensitive data generated and stored on smartphones such as personal information, financial transactions, and business communications [1]. Data safety and privacy have become major concerns [2]. To address these concerns, encryption is employed to secure data both in transit (when data is being transmitted over networks) and at rest (when stored on the device). Advanced Encryption Standard (AES) is a widely adopted symmetric encryption algorithm known for its strong security properties [3]. It has been approved by the National Institute of Standards and Technology (NIST) for securing sensitive data [1][3]. Due to its robustness and effectiveness, AES is commonly used on smartphones for various purposes, including encrypting files, securing communications, and protecting data stored in applications [5]. However, despite its high level of security, the use of AES on smartphones poses some challenges, particularly when dealing with large files in a resource-constrained environment. These challenges include

1 Poor performance issues: AES can be slow when encrypting large files, resulting in delays and a decrease in user experience. This can cause the smartphone to become sluggish, especially when the device is handling multiple encryption and decryption processes simultaneously.

2 Battery drain: The encryption and decryption processes in AES are computationally intensive. For smartphones, which are inherently limited by battery life, this can lead to rapid battery drainage, reducing the overall efficiency and usability of the device.

3 Resource utilization: Some Smartphones have limited processing power, memory, and storage compared to desktop computers and servers. Running AES on such devices, especially without optimization, can lead to high CPU usage, increased thermal output, and potential overheating, which impacts the device's performance and longevity.

This paper suggests an enhanced AES encryption algorithm that aims to solve the problem of balancing between performance and complexity in resource-constrained environments like smartphones. This is done by streamlining the algorithm to use fewer computational resources without compromising the security level.

The remainder of this study is structured as follows: Section 2 provides an overview of the existing AES algorithm, Section 3 reviews related work, Section 4 outlines the proposed methodology, Section 5 details the evaluation of the enhanced algorithm, Section 6 presents the results and discussion, and finally, the conclusion is provided in Section 7.

International Journal of Computer Applications Technology and Research
Volume 14–Issue 03, 132 – 139, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1403.1012

2. EXISTING AES ALGORITHM

The process of encryption in the AES algorithm consists of a set of steps or transformations that are carried out onto the data [7]. In AES 128 variant, it takes a block of 128- bits with 128- bits key and then performs four operations in ten rounds to get a cipher text. The operations include an initial add round key transformation, substitution byte, shift rows, mix-columns and add round key [8]. The output of one operation is the input of the next operation and the decryption process is the inverse of the encryption process.

2.1 AddRound key Transformation

The add round key is performed by XORing the plaintext with the round key,

State = Plaintext ⊕ Round Key [i] .

Where state – The current state matrix.

⊕ - Bitwise XOR operation.

Round key[i] – Round key derived for the i-th round using AES key schedule.

2.2 Substitution byte operation Transformation

During the substitution byte (sub bytes) step, each byte in the state is replaced with another byte using a fixed substitution box (S-Box) [9]. The AES substitution box (s-box) is a 16x16 matrix that contains all possible 256-byte values (from 0x00 to 0xFF). Each byte in the state is used as an index to look up its corresponding byte in the S-box.

2.3 Shift row operation Transformation

This step introduces diffusion by rearranging the bytes within the rows of the matrix [10]. Unlike sub bytes, the shift rows operation does not involve any complex mathematical transformations like substitutions or arithmetic operations. Instead, it relies on a simple shifting mechanism [5][11]. The rows of AES are shifted cylindrically using the following equation and steps to give new position for all element in i row.

S′i.j = Si, (j + i) mod 4

Where,

S is the original state matrix

S′ is the state matrix after the shift rows transformation

i represents the row index (starting from 0).

j represents the column index (starting from 0)

(j + i) mod 4 determines the new column position for each element in row i

Consider a matrix before applying AES shift row formula.

$$S = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Using the equation $S'_{i,j} = S_{i,\,(j+i) \bmod 4}$

Row 0: $S'_{0,j} = S_{0,\,(j+0) \bmod 4}$ - no shift (all elements remain in their original positions)

$S'_{0,0} = S_{0,0}$, $S'_{0,1} = S_{0,1}$, $S'_{0,2} = S_{0,2}$, $S'_{0,3} = S_{0,3}$

Row 1: $S'_{1,j} = S_{1,\,(j+1) \bmod 4}$ – elements shift left by 1 position

$S'_{1,0} = S_{1,1}$, $S'_{1,1} = S_{1,2}$, $S'_{1,2} = S_{1,3}$, $S'_{1,3} = S_{1,0}$.

Row 2: S′2.j = S2, (j + 2) mod 4 - elements shift left by 2 positions)

S′2,0 = S2, 2, S′2,1 = S2,3, S′2,2 = S2,0, S′2,3 = S2,1

Row 3: S′3.j = S3, (j + 3) mod 4 – elements shift left by 3 positions

S′3,0 = S3, 3, S′3,1 = S3,0, S′3,2 = S3,1, S′3,3 = S3,2.

The resulting position after the formular.

$$S' = \begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{11} & a_{12} & a_{13} & a_{10} \\ a_{22} & a_{23} & a_{20} & a_{21} \\ a_{33} & a_{30} & a_{31} & a_{32} \end{bmatrix}$$

2.4 Mix column operation Transformation

Mix columns step is a matrix multiplication of the state by Galois fields (GF) [13]. Each value in a column is eventually multiplied against every value of Galois matrix. The results of a column multiplication are XORed together to produce a ciphered column [8].

Each column of the state matrix is treated as a polynomial and multiplied with a fixed polynomial $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$ in GF (2^8)

For each column c = (s_0, s_1, s_2, s_3):

$$\begin{bmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

Multiplication is performed using polynomial arithmetic modulo $x^8 + x^4 + x^3 + x + 1$

In AES operation, mix columns are the highest computational burden of the four. There are two arithmetic operations in the mix column: multiplication and addition. Because of the complicated mathematical process that requires computing resources in a software implementation of AES, it is a costly transformation that slows down the encryption process.

International Journal of Computer Applications Technology and Research
Volume 14–Issue 03, 132 – 139, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1403.1012

2.5 Key expansion

Key expansion generates a series of round keys from the initial cipher key

Word (W): A word is a 32- bit portion of the key.

Rcon (round constant): Used in key expansion, define as Rcon(i)= $\{02^{i-1}, 00,00,00\}$ in hexadecimal

Steps:

Split the original key into 4 words (W_0, W_1, W_2, W_3).

Each new word is generated as:

$W_i = W_{i-4} \oplus$ Sub Word (Rot Word (W_{i-1})) $\oplus$ Rcon [i/4] for I = 4,8,12, …

Sub Word: Applies s-box substitution to each byte

Rot Word: Rotates the word left by one byte

Figure 1 shows the encryption and decryption process of AES.

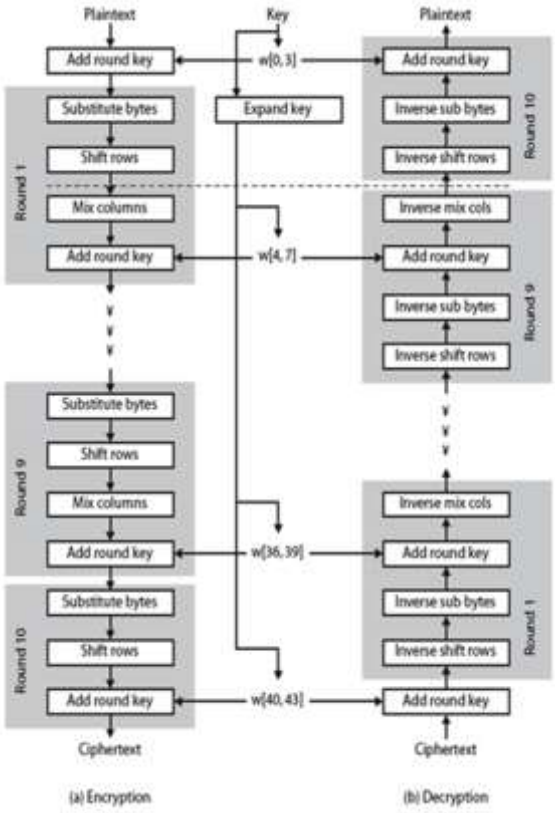

Figure 1, Encryption and decryption process of AES [14].

3. RELATED WORKS

In today's resource-constrained environments, the focus is increasingly shifting toward efficient cryptographic algorithms. Several studies have proposed modifications of AES to reduce complexity, improve security, execution speed, memory efficiency, and adaptability for resource-constrained environments such as smartphones and IoT devices.

[2] utilized the linear feedback shift register (LFSR) for key generation and reduced the number of rounds to five, resulting in faster encryption time; however, the impact on security was not specified. [3] integrated AES with the flower pollination algorithm to enhance the S-Box, resulting in improved security and increased randomness in key generation. [4] implemented a flexible AES variant, transitioning between AES-128 and AES-256, incorporating a secure key generator (SKG) and one-time pad (OTP), which enhanced security adaptability based on requirements. Proposed modified AES proposed in [5] tailored for IoT applications, incorporating dynamic shift rows and initial permutation, which resulted in a 19.24% higher throughput compared to related studies. [6] introduced a preprocessing stage called zigzag padding, removed the Sub Byte operation, and reduced the number of rounds. These modifications led to a 10% increase in encryption throughput, a 9.3% increase in decryption throughput, and a 29.5% improvement in randomness. Similar to [6],[7] reduced the number of rounds to six, resulting in improved security and speed, with encryption throughput increasing by 10% and decryption by 9.3%. [8] eliminated the Mix Columns operation, directly obtaining output from Shift Rows, which led to faster execution and reduced complexity.[9] reduced the Mix Columns operation and introduced a continued fraction transformation to compress the plaintext to 16 bits. Their approach achieved an execution time of 280ms for 45.1KB, compared to 294ms in standard AES. [10] modified the Shift Rows and Mix Columns operations, integrating them with Add Round Key into a single cycle while reducing the number of rounds to six. These enhancements resulted in a faster implementation with improved security. [11] replaced the Mix Columns operation with chaotic theories, achieving the shortest encryption time of 0.0169s, compared to 4.1249s for the SPECK algorithm. [12] modified the Sub Byte and Shift Rows operations, achieving a 10% faster performance than the original AES, with an avalanche effect of 52.08% compared to 51.82% in the original AES. [13] replaced the standard Shift Rows with a dynamic Shift Rows and utilized two S-Boxes instead of one. Their approach achieved an encryption time of 0.0169s, outperforming the SPECK algorithm. [14] replaced the Mix Columns operation with Junction Jump, reduced the number of rounds, and achieved a 14.68% decrease in memory usage along with a 50% reduction in encryption and decryption time.

4. PROPOSED METHODOLOGY

This section presents the methodology adopted in this study in detail.

4.1 Modified Sub Bytes Transformation

In our modified shift row, we provided a more complex transformation where elements in matrix both rows and columns are mixed, this will spread diffusion more than the conventional AES. We derived a formula that determines the new position of byte after rows and columns are mixed.

Given that the new position $M'_{[i', j']}$ for an element $M_{[i, j]}$ should depend on the function of both the original row i and column j indices.

The formula can be expressed as

$$M'_{[i', j']} = M_{[(i + j) \bmod 4, (j + 2i) \bmod 4]} \quad [3]$$

Where

M is the original input matrix.

M′ is the transformed output matrix

i refer to the element in the original matrix M

j refer to the column index of the element in the original matrix M

i′ and j′ defines the location of the element in the transformed matrix M′.

International Journal of Computer Applications Technology and Research
Volume 14–Issue 03, 132 – 139, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1403.1012

Mod 4 ensures that the indices stay within the valid range for a 4x4 matrix (0,1, 2, 3)

Consider a matrix before applying our derived dynamic shift byte equation.

$$M = \begin{bmatrix} M[0,0] & M[0,1] & M[0,2] & M[0,3] \\ M[1,0] & M[1,1] & M[1,2] & M[1,3] \\ M[2,0] & M[2,1] & M[2,2] & M[2,3] \\ M[3,0] & M[3,1] & M[3,2] & M[3,3] \end{bmatrix}$$

Apply Formula: $M'_{[i', j']} = M_{[i + j) \bmod 4, (j + 2i) \bmod 4]}$

Row 0

M′[0,0] = M [(0 + 0) mod 4, (0 + 2 x 0) mod 4] = M [0 ,0]

M′[0,1] = M [(0 + 1) mod 4, (1 + 2 x 0) mod 4] = M [1 ,1]

M′[0,2] = M [(0 + 2) mod 4, (2 + 2 x 0) mod 4] = M [2 ,2]

M′[0,3] = M [(0 + 3) mod 4, (3 + 2 x 0) mod 4] = M [3 ,3]

Row 1

M′[1,0] = M [(1 + 0) mod 4, (0 + 2 x 1) mod 4] = M [1 ,2]

M′[1,1] = M [1 + 1) mod 4, (1 + 2 x 1) mod 4] = M [2 ,3]

M′[1,2] = M [(1 + 2) mod 4, (2 + 2 x 1) mod 4] = M [3 ,0]

M′[1,3] = M [(1 + 3) mod 4, (3 + 2 x 1) mod 4] = M [0 ,1]

Row 2

M′[2,0] = M [(0 + 0) mod 4, (0 + 2 x 2) mod 4] = M [2 ,0]

M′[2,1] = M [2+ 1) mod 4, (1 + 2 x 2) mod 4] = M [3 ,1]

M′[2,2] = M [(2 + 2) mod 4, (2 + 2 x 2) mod 4] = M [0 ,2]

M′[2,3] = M [(2 + 3) mod 4, (3 + 2 x 2) mod 4] = M [1 ,3]

Row 3

M′[3,0] = M [(3 + 0) mod 4, (0 + 2 x 3) mod 4] = M [3 ,2]

M′[3,1] = M [3 + 1) mod 4, (1 + 2 x 3) mod 4] = M [0 ,3]

M′[3,2] = M [(3 + 2) mod 4, (2 + 2 x 3) mod 4] = M [1 ,0]

M′[3,3] = M [(3 + 3) mod 4, (3 + 2 x 3) mod 4] = M [2 ,1]

The resulting position after the formula.

`

$$M' = \begin{bmatrix} M[0,0] & M[1,1] & M[2,2] & M[3,3] \\ M[1,2] & M[2,3] & M[3,0] & M[0,1] \\ M[2,0] & M[3,1] & M[0,2] & M[1,3] \\ M[3,2] & M[0,3] & M[1,0] & M[2,1] \end{bmatrix}$$

The results of the two methods, namely the AES ShiftRows and our dynamic shift-byte approach, indicate that our approach achieves greater diffusion by mixing the entire byte across the matrix, compared to the conventional AES approach, which only mixes within individual rows.

4.2 New Sub Byte Transformation

The output of the sub-byte and dynamic mix byte presents as 4x4 matrix in hexadecimal respectively, XORed and the output is the new mix column. bitwise operation XOR was introduced because it is effective and has less complex computation. It will replace the mix column operation, that was eliminated due to its long and time-consuming mathematical operations.

Mathematically, the formula is expressed as

N [i, j] = S [i, j] ⊕ D [i, j]. [4]

Where,

S[i j] - Is the input byte from the sub byte state matrix

⊕- The XOR (exclusive OR) operation, a fundamental binary operation that outputs 1 if the bits are different and 0 if they are the same.

D[i, j]- Is the input byte from the dynamic mix byte state matrix

N [i, j] – Is the final byte after XOR operation i.e. the new sub byte. Figure 2 shows the encryption and decryption process of E-AES.

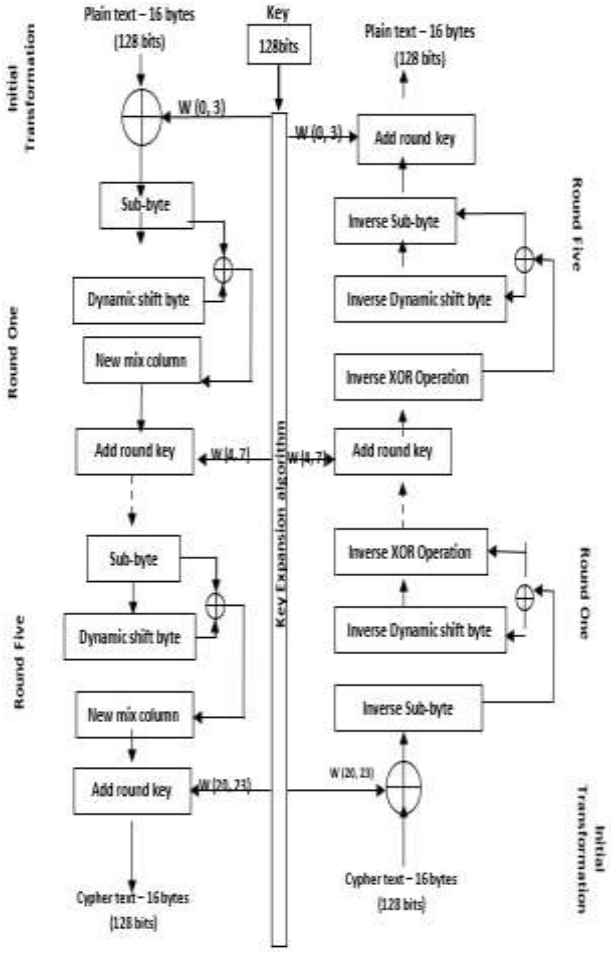

Figure 2. Encryption and decryption process of AES

5. EVALUATING THE PERFORMANCE OF THE TWO ALGORITHMS

The enhanced AES and conventional AES were assessed based on the avalanche effect, encryption/decryption time, and memory usage. The evaluation was carried out on a smartphone with the following specifications: Dolphin v7.6.0,

International Journal of Computer Applications Technology and Research
Volume 14–Issue 03, 132 – 139, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1403.1012

Android 11, 2GB RAM, and a non-removable 5000mAh battery, with both algorithms installed.

5.1 Measuring the avalanche effect

The avalanche effect is a desirable property of block ciphers that ensures a single bit flip in input text produces at least 50% change in the output text. If a cryptographic algorithm does not exhibit a significant degree of avalanche effect (at least 50%), then that algorithm has poor randomization. Thus, cryptanalysts can make predictions about the input, only being given the output. This may be enough to partially or completely break the algorithm.

Avalanche effect could also be computed using Equation (5) such that:

$$\text{Avalanche Effect} = \frac{\text{No bits differs in two cipher text}}{\text{Total no of bits in cipher}} X100\%$$

The avalanche effect of the enhanced AES and conventional AES were carried out using a short plain text. A short plain text: I Love UNIZIK with hexadecimal values: 49 20 6c 6f 76 65 20 55 6e 69 7a 69 6b was used to measure the avalanche effect of the enhanced and conventional AES.

The following steps were used to calculate the avalanche effect of the selected algorithms.

1. Encrypt plaintext P1 using key K1 to produce ciphertext C1.
2. Flip one bit in the plaintext (or key) to create P2 (or K2).
3. Encrypt the modified plaintext (or key) to produce ciphertext C2
4. Convert C1 and C2 into binary format.
5. Count the number of differing bits between C1 and C2.
6. Calculate Avalanche Percentage using the formula

Table 1 presents the avalanche effect that was obtained after flipping a single bit into the plain text. Table 1 shows the result of the avalanche effect. Figure 3 illustrates the result of the avalanche effect in a bar chart.

5.2 Measuring the encryption and decryption time

Encryption time is the time taken to convert a plain text to a cipher text and the time that is needed to convert the cipher text back to the plain text the decryption time. The encryption time and decryption time are expected to be small to have a responsive and fast system. Furthermore, they depend to some extent on the configuration of the system used.

Table 2 presents the encryption and decryption time test results in milliseconds (ms), which was obtained by computing the average encryption/decryption time after encrypting/decrypting the same input text while using the same key five times. Figure 4 shows the result comparison for encryption time for conventional AES and E-AES while figure 5 shows the result comparison for decryption time for conventional AES and E-AES.

5.3Measuring memory usage

To compare the memory usage of the two algorithms, i.e. the conventional AES and E-AES, we encrypted 1000kb with the different algorithm using the same key. To do this, a popular AES android-based application called Android crypt was installed on our device to evaluate the memory usage of both algorithms. Qualcomm trepn profiler was used to profile the memory for the two encryption applications. Trepn profiler is a diagnostic tool for profiling performance and power consumption on android applications running on devices. Figure 6 presents the result of memory usage.

6 RESULTS AND DISCUSSION

This section presents the findings of this research

Table 1. Avalanche effect result obtained after flipping a single bit in the plaintext.

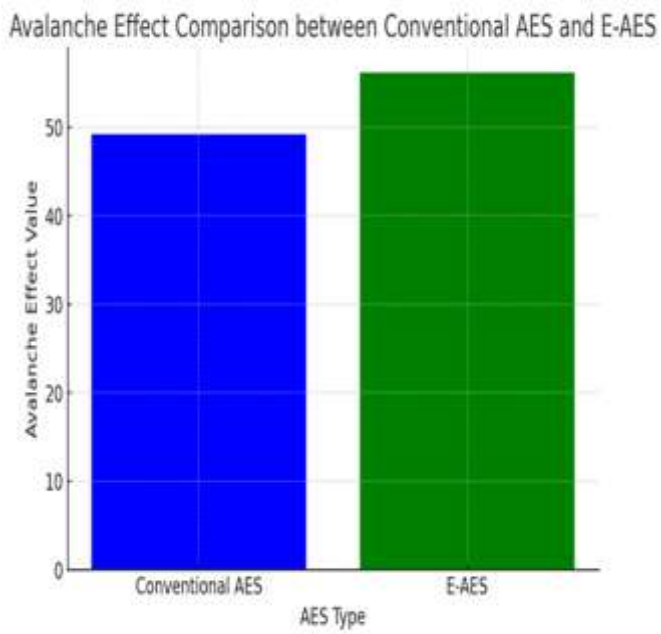

Figure 3 Result of the avalanche effect

From the result, the enhanced AES achieved an avalanche effect of 56.25% when compared to 49.21875% achieved by the conventional AES algorithm. This characteristic is essential for cryptographic strength, as it helps obscure patterns and resists differential attacks, indicating E-AES offer better security than conventional AES

International Journal of Computer Applications Technology and Research
Volume 14–Issue 03, 132 – 139, 2025, ISSN:-2319–8656
DOI:10.7753/IJCATR1403.1012

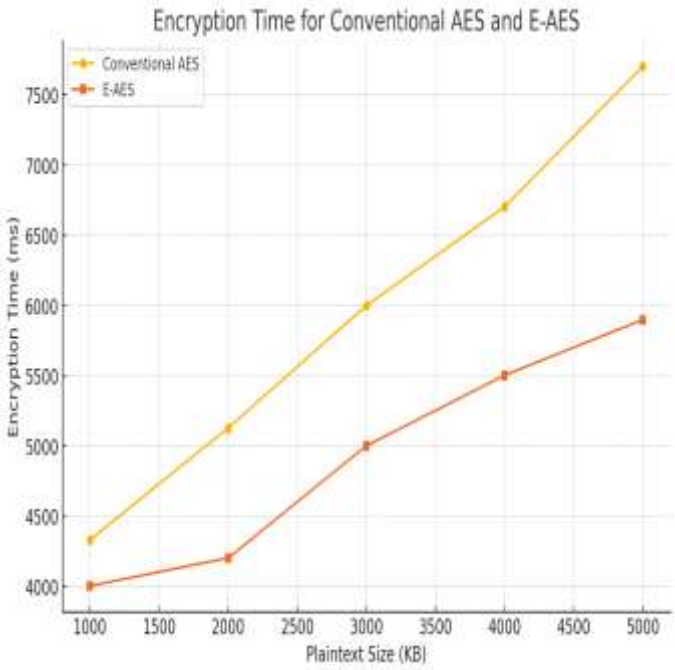

Figure 4 Result comparison of encryption time of conventional AES and E-AES.

Table 1. Avalanche effect result obtained after flipping a single bit in the plaintext.

Algorithm	Secret key	Plaintext	Plaintext (Hex)	Ciphertext (Hex)	Avalanche effect
Conventional AES	2b7e151628ae 2a6abf7158809cf4f3c	I love Unizik	49 20 6c 6f 76 65 20 55 6e 69 7a 69 6b	3925841d02 dc09fbdc11859 7196a0b32	0.492187 (49.2187)
		I love Unizic	49 20 6c 6f 76 65 20 55 6e 69 7a 69 63	3243f6a888 5a308d31319 8a2e0370735	
E-AES		I love Unizik	49 20 6c 6f 76 65 20 55 6e 69 7a 69 6b	3925841d02 dc09fbdc118 597196a0b32	0.5625 (56.25)
		I love Unizic	49 20 6c 6f 76 65 20 55 6e 69 7a 69 63	2a1793732f6 7437d447f8d 55ed40a50e	

Table 2 Encryption and decryption Time result

Plain Text Size (kb)	Algorithm	Average encryption time (MS)	Average decryption time (MS)
1000 kb	Conventional AES	4332	4337
	E-AES	4005	4027

2000 kb	Conventional AES	5130	5138
	E-AES	4204	5003
300kb	Conventional AES	6001	6007
	E-AES	5003	5888
400kb	Conventional AES	6705	7148
	E-AES	5505	6902
500kb	Conventional AES	7704	8008
	E-AES	5900	7900

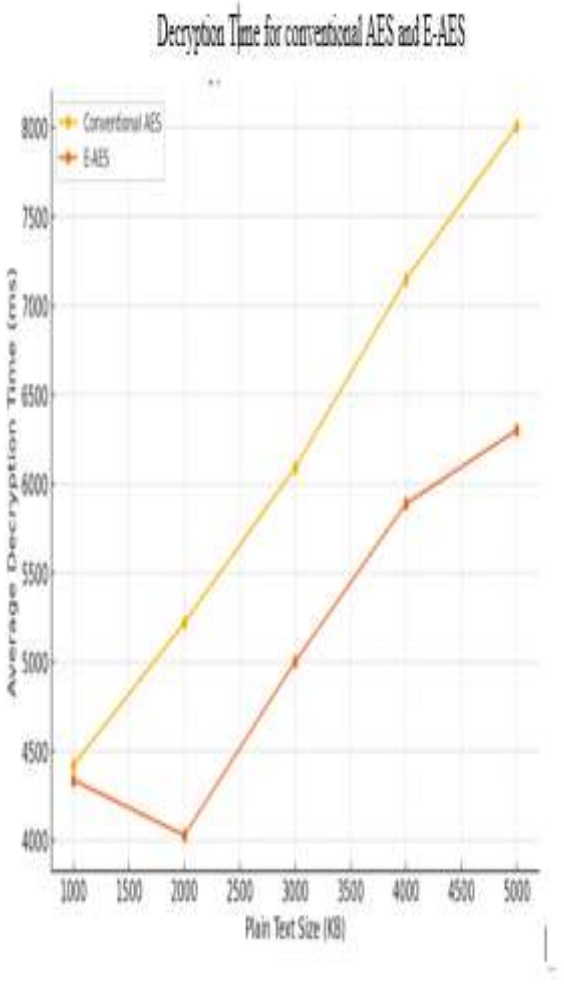

Figure 5. Result comparison of decryption time of conventional AES and E-AES

The encryption and decryption result above indicates that the enhanced AES has a slight increase in the encryption and decryption time when compared to the conventional AES algorithm. The graph highlights E-AES as a more efficient choice than Conventional AES for encryption, especially for larger plaintext sizes. This efficiency can contribute to reduced energy consumption and faster performance in data-intensive applications

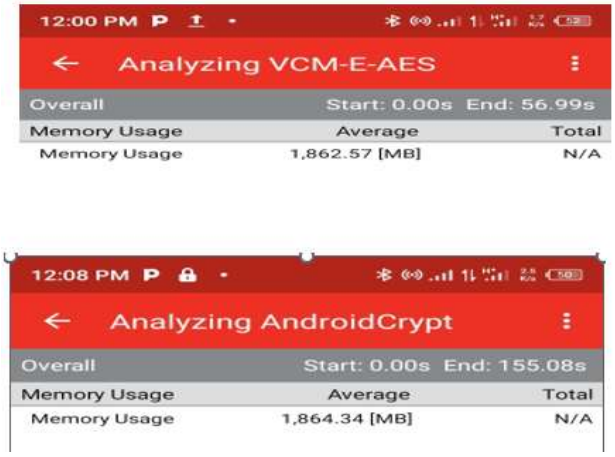

Figure 6. Result of memory usage

The result as shown in figure 4 demonstrates our application keep maintaining the smallest memory usage when encrypted 1000KB file. E-AES used an average space of 1862.57(MB) when encrypting 1000KB while Conventional AES based application used an average space of 1864.34(MB). This means in every 1000KB encrypted our application saves 1.77(MB) of memory space.

7 CONCLUSION

The Enhanced AES (E-AES) cryptographic model outperforms conventional AES across key metrics, including a higher avalanche effect, faster encryption and decryption times, and reduced memory consumption. These enhancements not only improve energy efficiency, but contribute to longer battery life strength, it improved also security which is an essential factor in mobile device encryption. This study affirms E-AES as a robust, secure, and efficient solution for data protection in mobile environments, addressing complexity, and performance.

REFERENCES

[1] Sun, X. and Shanshan, L. (2023). Multi-sensor network tracking research utilizes searchable encryption algorithm in the cloud computing environment. *International Journal Sensor Networks*, 2024 (7), 1-9.

[2] Ali, H.H. and Shaimaa, H.S. (2020). Modified advanced encryption standard algorithm for fast transmitted data protection. *In the 2nd International scientific conference of AI- Ayen university*, 6-11 October 2024 (pp. 1-11). Melbourne, Victoria, Australia: IEEE Xplore

[3] Indrasena, M. and Kumar, S. (2020). A modified advanced encryption standard. *Journal of Mechanics of Continua and Mathematical Science*, 4(5), 333–340.

[4] Shikha, G., Satbiri, J., Mohit, A. and Nalin, N. (2024). An encryption approach to improve the security and performance of data by integrating AES with a modified OTP technique. International Journal of Advanced Intelligence Paradigms,27(2), 129-149.

[5] Fadhil, Meryam Saad, Alaa Kadhim Farhan, and Mohammad Natiq Fadhil. (2021).

[6] Gao, R., Shen, L., Yuqi, G. and Rui, G. (2021). A lightweight cryptographic algorithm for the transmission of images from road environments in self-driving. Journal of cybersecurity 4(1), 402-407

[7] Abdul, H., Farah T., Abdul, Monem, S.R. and Hala B.A. (2021). A secure environment using a new lightweight AES encryption algorithm for e-commerce websites. security and communication networks. International Journal of Research and Innovation. 56(3), 456-462

[8] Kumar, K., Ramkumar, K. and Amanpreet, K. (2022). A lightweight AES algorithm implementation for encrypting voice messages using field programmable gate arrays. Journal of King Saud University - Computer and Information Sciences, 34(6), 3878–3885.

[9] Mohammad, H.M. and Abdullah, A. (2022). Enhancement Process of AES: a lightweight cryptography algorithm-AES for constrained devices. Telkomnika Telecommunication Computing Electronics and Control 20(3), 551–560.

[10] Rasool, S.S, Alaa, K.F. and Ali, S. (2022). Lightweight modifications in the advanced encryption standard (AES) for IOT application, a comparative survey. In International Conference on Computer Science and Software Engineering (CSASE), 20-23 November 2013 (pp.612-617). Rwanda: IOP conference series.

[11] Jassim, S. and Farhan, A. (2022). Designing a new lightweight AES algorithm to Improve the Security of the IoT Environment. Iraqi Journal of Computers, Communications, Control, and Systems Engineering, 22(2), 96–108

[12] Hammod, D.N. (2022). Modified lightweight AES based on replacement table and chaotic system. International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). 43(2), 1-5

[13] Sameeh, A.J. and Alla, K.F. (2022). Designing a new lightweight AES algorithm to improve the security of the IoT environment. Iraqi Journal of Computer, Communication, Control and System Engineering, 8(9) 96–108.

[14] Shah, A., Sanja,Y S., Hiren, P., and Namit, S. (2023). LSA: A lightweight symmetric encryption algorithm for resource constrained IOT system. Journal of Reliability: Theory and Application, 18(3),374-379