Improvisation of the Advanced Encryption Standard Algorithm (AES) By Mapping the S-Box on the Mixcolumns Modification

Aminudina) and Bashor Fauzan Muthohirinb)

Department of Informatics Engineering, Universitas Muhammadiyah Malang, Malang, Indonesia

b) Corresponding author: [bashorfauzan@umm.ac.id](mailto:bashorfauzan@umm.ac.id.com)

a) aminudin2008@umm.ac.id

**Abstract.** AES has a very good confusion lookup table to break non-linear patterns. Diffusion changes one bit of the input and changes part of the output to the average value. This process is because the secret key has been mixed up in each part, so that the attacker cannot predict the initial key calculation before encryption. Based on the research that has been conducted, several weaknesses of the AES algorithm have been found, namely the difficulty of key management because each data communication requires a different key, the symmetric key that is used repeatedly allows for key leaks over a long period of time, and referring to the mathematical basis of GF(2^8) if the mathematical equation is solved then the security sequence can be destroyed. In addition to brute force, there are several attacks that can be launched to break the AES defense. Some possible types of attacks are differential cryptanalysis and linear cryptanalysis, truncated differential, and the square attacks and interpolation attacks. The latest attack that has been proven to be able to completely destroy the AES security system is biclique cryptanalysis. This research modified the sbox lookup table and constant matrix on mixcolumns. The improvisation was done with the desire to be able to improve performance to be more efficient and processing speed. AES is more improved. Tests conducted on memory usage, computing time and avalanche effect percentage each have a difference of 24mb, 18.9 seconds, and an increase in avalanche effect of 0.92%. Based on these data, it can be seen that the performance of this AES improvisation has been able to improve algorithm performance by reducing time and memory and increasing the percentage of avalanche effect.

**Keywords:** AES, diffusion, lookup, brute force, avalanche effect

# INTRODUCTION

Cryptography is the art of securing information in the form of digital data, whether text, images or other forms of data. The mechanisms that have developed in cryptographic technology are: Hash Function, symmetric key encipherment, and asymmetric key encipherment. AES (Advanced Encryption Standard) is one of the symmetric key encryption algorithms in the block cipher category. AES is able to secure data from brute force attacks because it is a symmetric algorithm with a key length of 2^128 [1]. It would take about 10^10 years with a billion processors to crack it. So, only those who have the private key can enter the data or system [2].

Good cryptographic principles will make the statistical relationship between plaintext, ciphertext and key very complex, making it difficult for cryptanalysts to find attack patterns [3]. According to Claude Shannon, there are two important properties that must be considered in a secure cipher, namely confusion and diffusion. William Stalling in his book "Cryptography and Network Security Principal" explains that the concept of confusion is a statistical structure of plaintext that is made randomly irregular into statistics with a long range of ciphertext [4]. The principle is obtained by each plaintext digit affecting the value of many ciphertext digits, in general and related that each ciphertext digit is affected by the number of plaintext digits. Confusion makes the relationship between statistics of ciphertext and encryption key values ​​as complex as possible by using complex substitution algorithms [5]. AES has a very good confusion lookup table to destroy non-linear patterns. Diffusion changes one input bit and changes some of the outputs to the average value. This process is because the secret key has been mixed up in each part, so that the attacker cannot predict the initial pre-encryption key calculation.

Based on comparative research conducted by Muttaqin et.al, several weaknesses of the AES algorithm have been found, namely the difficulty of key management because each data communication requires a different key, symmetric keys that are used repeatedly allow for key leaks over a long period of time, and referring to the mathematical basis of GF(2^8) if the mathematical equation is solved, the security sequence can be destroyed [6]. In addition to brute force, there are several attacks that can be launched to destroy the AES defense. Some types of possible attacks are differential cryptanalysis and linear cryptanalysis, truncated differential, and the square attacks and interpolation attacks. The latest attack that has been proven to be able to completely destroy the AES security system is biclique cryptanalysis. Research conducted by Igor Bony states that the lookup table in AES has both strengths and weaknesses because it makes operations less complicated [7]. Based on previous research, improvisation is needed to improve algorithm execution performance as well as increase security from attacks.

The block cipher structure in AES has four transformations that can increase confusion and diffusion. The Addroundkey, Shiftrows, Subbytes transformations have confusion properties, and mixcolumns have diffusion properties in their transformation algorithms. This research will improvise on two transformations, namely Subbytes and mixcolumns to improve security performance. Research conducted by Ericky Benna has improvised on mixcolumns by using matrix transpose operations. The improvisation he made has been able to increase the value of the avalanche effect.

# METHODS

Cryptographic algorithms basically operate following the nature of digital computer operations, namely data processing against binary symbols. As mentioned by William Stalling, basically the polynomial algebra used in AES includes multiplication, addition, inverse and substitution. The polynomial algebra used in AES operations is a finite field or Galois field of degree 4 which is denoted by the equation (1). The units used in AES include states divided into 4x4 matrix blocks measuring 16 bytes or 128 bits. Each block consists of four words measuring 4 bytes or 128 bits. The notation units in AES are influenced by the equation (1) which if described in polynomial operations will result in 128 bits as a binary operation in computing.

(1)

AES basically consists of four transformations, namely Addroundkey, Shiftrows, Subbytes, and Mixcolumns. Each transformation is interconnected in a repeating stage based on the key length. A key with a length of 128 bits is equivalent to 16 bytes, will produce a 128-bit AES algorithm with 10 iterations containing the four transformations. The AES encryption and decryption cryptographic algorithm can be seen in **FIGURE 1**.

A diagram of a computer program

Description automatically generated

**FIGURE 1**. Encryption and decryption of AES 128 bit

## AES Standard

Addroundkey is a transformation process of mixing two bytes of data from plaintext with a key. Basically algebraically, the process in addroundkey uses the basis of finite field multiplication. The exclusive or operation is used as an operator in mixing value state plain ​​and keys as in the equation (2). Addroundkey in the next round mixes data between the key schedule and the state-plain.

 (2)

AES as a block cipher cryptography algorithm, the operation used is a matrix consisting of 4 words that become a single state unit of 16 bytes. Shiftrows transformation is the process of shifting the rows of the state-plain matrix. Row r = 1 is shifted by 1 byte, r = 2 is shifted by 2 bytes, r = 3 is shifted by 3 bytes, while row r = 0 does not experience a shift process. The mathematical basis of shiftrows is cyclic permutations. Cyclic permutations on shiftrows in the equation (3) which then produces the equation (4) as a good state-plain matrix from the results of row shifts.

(3)

Subbytes is a transformation that involves a substitution process with a lookup table called an s-box. Basically, operations on subbytes use polynomial operations as shown in the equation (5). The use of lookup table as a development to simplify the transformation process. The s-box lookup table can be seen in **TABLE 1**. Substitution is done based on the stateplain index value which is then used to get a new value based on the similarity of the row and column indexes in the new state-plain.

**TABLE 1.** Lookup table s-box

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
| **0** | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 1 | 67 | 2B | FE | D7 | AB | 76 |
| **1** | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| **2** | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| **3** | 4 | C7 | 23 | C3 | 18 | 96 | 5 | 9A | 7 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| **4** | 9 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| **5** | 53 | D1 | 0 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| **6** | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 2 | 7F | 50 | 3C | 9F | A8 |
| **7** | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| **8** | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| **9** | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| **A** | E0 | 32 | 3A | 0A | 49 | 6 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| **B** | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 8 |
| **C** | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| **D** | 70 | 3E | B5 | 66 | 48 | 3 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| **E** | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| **F** | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

Matrix multiplication between the two stateplain and constant matrix is ​​used in mixcolumns transformation. The concept of diffusion mentioned by Stalling in AES is found in mixcolumns transformation. The EBC operation that unites data bits and replaces them with new values ​​results in the distribution of data in the ciphertext being more complicated to determine its pattern. The multiplication used in the transformation cannot be separated from the rules of polynomial multiplication which uses the exclusive OR (XOR) operator. The mixcolumns transformation equation is presented in the equation (6).

 (5)

## AES Modification

The improvisation carried out in this study lies in the development of subbytes and mixcolumn transformations. Overall, the subalgorithms of both transformations do not experience changes, but improvisation is carried out on the s-box lookup table and constant matrix. Substitution in the subbytes transformation involves a lookup table that will become a new value in the stateplain. Improvisation by multiplying the values ​​in each index has been carried out in previous studies. The standard s-box value is a hexadecimal number that is multiplied by a hexadecimal number with values ​​02 and 03. In contrast, this study only does one improvisation, namely the multiplication of 02 lookup table s-box. Based on the results of the multiplication of each index value, some produce values ​​of more than 8 bits so that additional operations are used to reduce them back to 8 bits according to the basic rules of finite fields. The s-box index value reduction operation can use the xor operation with 11B or use a polynomial operation as in the equation (7). After all the values ​​are 8 bits in size, you can see the modified s-box index value becomes like in **TABLE 2**.

(6)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **A** | **B** | **C** | **D** | **E** | **F** |
| **0** | C6 | F8 | EE | F6 | FF | D6 | DE | 91 | 60 | 1 | CE | 56 | E7 | B5 | 4D | EC |
| **1** | 8F | 1F | 89 | FA | EF | B2 | 8E | FB | 41 | B3 | 5F | 45 | 23 | 53 | E4 | 9B |
| **2** | 75 | E1 | 3D | 4C | 6C | 7E | F5 | 83 | 68 | 51 | D1 | F9 | E2 | AB | 62 | 2A |
| **3** | 8 | 95 | 46 | 9D | 30 | 37 | 0A | 2F | 0E | 24 | 1B | DF | CD | 4E | 7F | EA |
| **4** | 12 | 1D | 58 | 34 | 36 | DC | B4 | 5B | A4 | 76 | B7 | 7D | 52 | DD | 5E | 13 |
| **5** | A6 | B9 | 0 | C1 | 40 | E3 | 79 | B6 | D4 | 8D | 67 | 72 | 94 | 98 | B0 | 85 |
| **6** | BB | C5 | 4F | ED | 86 | 9A | 66 | 11 | 8A | E9 | 4 | FE | A0 | 78 | 25 | 4B |
| **7** | A2 | 5D | 80 | 5 | 3F | 21 | 70 | F1 | 63 | 77 | AF | 42 | 20 | E5 | FD | BF |
| **8** | 81 | 18 | 26 | C3 | BE | 35 | 88 | 2E | 93 | 55 | FC | 7A | C8 | BA | 32 | E6 |
| **9** | C0 | 19 | 9E | A3 | 44 | 54 | 3B | 0B | 8C | C7 | 6B | 28 | A7 | BC | 16 | AD |
| **A** | DB | 64 | 74 | 14 | 92 | 0C | 48 | B8 | 9F | BD | 43 | C4 | 39 | 31 | D3 | F2 |
| **B** | D5 | 8B | 6E | DA | 1 | B1 | 9C | 49 | D8 | AC | F3 | CF | CA | F4 | 47 | 10 |
| **C** | 6F | F0 | 4A | 5C | 38 | 57 | 73 | 97 | CB | A1 | E8 | 3E | 96 | 61 | 0D | 0F |
| **D** | E0 | 7C | 71 | CC | 90 | 6 | F7 | 1C | C2 | 6A | AE | 69 | 17 | 99 | 3A | 27 |
| **E** | D9 | EB | 2B | 22 | D2 | A9 | 7 | 33 | 2D | 3C | 15 | C9 | 87 | AA | 50 | A5 |
| **F** | 3 | 59 | 9 | 1A | 65 | D7 | 84 | D0 | 82 | 29 | 5A | 1E | 7B | A8 | 6D | 2C |

**TABLE 2** Modified s-box table

The constant matrix in the mixcolumns transformation is a 4x4 matrix. Previous research has conducted a matrix transpose mechanism with the aim of increasing the percentage of the avalanche effect. Transpose on the constant matrix is ​​expected to be able to increase data distribution as diffusion in AES improvisation. The next constant matrix modification produces the equation (7).

 (7)

In terms of the stages of the mixcolumns transformation operation as a multiplication operation of the two matrices, there is no change. The value of the constant matrix that has been changed can affect the results of the operation which becomes the value of the new stateplain and will be used in the next transformation stage. The results of the modification of the two transformations are expected to be able to improve the performance of AES, both confusion, diffusion, and device efficiency.

# RESULTS AND DISCUSSION

Implementation is done by creating a program based on the description of the AES formula and transformation. The data sample is then processed in both implementation programs of the standard AES and modified AES. The results of the tests from the two algorithms are then compared and analyzed so that the results and conclusions of the research are obtained.

## TESTING USING AVALANCHE EFFECT

**TABLE 3.** Standard AES avalanche effect test results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | File Name | Size | Number of bits | Bit Changes | Avalanche |
|  | (.docx) | (Bytes) | (bit) | (bit) | Effect (%) |
| 1 | 01A | 13.187 | 560 | 304 | 54,29 |
| 2 | 02A | 13.550 | 2.304 | 1.176 | 51,04 |
| 3 | 03A | 14.590 | 3.200 | 1.649 | 51,53 |
| 4 | 04A | 15.764 | 3.600 | 1.824 | 50,67 |
| 5 | 05A | 15.905 | 4.480 | 2.259 | 50,42 |
| Average | | | 2.828 | 1.485 | 52,51 |

Based on **TABLE 3**, it can be seen that the avalanche effect results on the standard AES average 52.51%. The highest percentage is found in files with the smallest size, while the percentage of other data sample sizes has an average of 50%. Based on the randomness of the data generated in the AE calculation, it shows that the size of the file does not affect the percentage of data randomness. So if seen based on the data distribution pattern, the ciphertext has a large confusion and diffusion nature. In **TABLE 4** show the result of the modified AES avalanche effect test.

**TABLE 4.** Results of the modified AES avalanche effect test

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | File Name | Size | Number of bits | Bit Changes | Avalanche |
|  | (.docx) | (Bytes) | (bit) | (bit) | Effect (%) |
| 1 | 01A | 13.187 | 560 | 313 | 55,89 |
| 2 | 02A | 13.550 | 2.304 | 1.203 | 52,21 |
| 3 | 03A | 14.590 | 3.200 | 1.705 | 53,28 |
| 4 | 04A | 15.764 | 3.600 | 1.884 | 52,33 |
| 5 | 05A | 15.905 | 4.480 | 2.324 | 51,87 |
| Average | | | 2.828 | 1.485 | 52,51 |

The percentage of avalanche effect on modified AES tends to be higher than standard AES. As the average of the two algorithms has a percentage difference of 0.92% greater than modified AES. This is influenced by the change in the modified s-box value so that it can provide a higher level of data bit distribution. The greater the percentage of AE, the greater the level of data security based on the confusion and diffusion of the algorithm.

## COMPUTE TIME TESTING

**TABLE 5.** Results of encryption computation time comparison

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | File Name | Size | Standard | Modification |
|  | (.docx) | (Bytes) | (ms) | (ms) |
| 1 | 01A | 13.187 | 560 | 116 |
| 2 | 02A | 13.550 | 2.304 | 160 |
| 3 | 03A | 14.590 | 3.200 | 265 |
| 4 | 04A | 15.764 | 3.600 | 1.049 |
| 5 | 05A | 15.905 | 4.480 | 1.693 |
| Average | | | 2.828 | 656 |

Based on the comparison results of the two encryption processes in **TABLE 5**, there is a significant change in execution time. The difference between the two algorithms has an average of 18,249ms greater than the standard AES. Based on the analysis of the performance measurement of the time variable in **TABLE 6**, the time efficiency of the modified AES can be said to be successful because it can reduce the computation time when executing data sample encryption.

**TABLE 6.** Computation time comparison test results description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | File Name | Size | Standard | Modification |
|  | (.docx) | (Bytes) | (ms) | (ms) |
| 1 | 01A | 13.187 | 2.261 | 141 |
| 2 | 02A | 13.550 | 8.238 | 280 |
| 3 | 03A | 14.590 | 12.779 | 375 |
| 4 | 04A | 15.764 | 33.496 | 1.197 |
| 5 | 05A | 15.905 | 61.282 | 2.101 |
| Average | | | 23.612 | 818 |

The performance of the algorithm can be considered based on the security of the data during computing, namely the difference in encryption and decryption time. The computation time of decryption is recommended to be greater than encryption. The lower the computation time of decryption, the greater the possibility of an attack by cryptanalysts to break the key and plaintext. The time difference of each algorithm during the encryption and decryption process has a larger time difference in decryption as in tables 3 and 4. Overall, the comparison of the computation time of the two algorithms has a fairly large average difference. The difference that occurs is due to changes in the scheme in the mixcolumns transformation.

In general, the method used in mixcolumns is 4x4 matrix multiplication. The shift left operator is used when the value of the constant matrix is ​​02 or 03. The implementation of the mixcolumn modification still uses the matrix multiplication rule but reduces one of the shift left operations. The diffusion change still increases in the modified AES because the data mixing rule to increase diffusion is still used by changing the constant matrix as in the equation (8). So that time efficiency and increased diffusion can improve the performance of the modified AES.

## TESTING USING MEMORY PROFILING

**TABLE 7.** Memory profiling of standard and modified AES encryption

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | File Name | Size | Heap used | |
| Standard | Modification |
|  | (.docx) | (Bytes) | (ms) | (ms) |
| 1 | 0 | 0 | 11.613.416 | 10.950.208 |
| 2 | 01A | 13.187 | 21.606.680 | 19.779.600 |
| 3 | 02A | 13.550 | 40.421.128 | 28.770.400 |
| 4 | 03A | 14.590 | 50.510.984 | 42.560.176 |
| 5 | 04A | 15.764 | 81.592.616 | 63.202.640 |
| 6 | 05A | 15.905 | 102.240.100 | 88.456.840 |
| Average | | | 61.596.984 | 34.821.761 |

**TABLE 7** shows the results of memory profiling by measuring the amount of heap used during program execution. This test analysis is carried out based on the amount of heap used when the program is run without being given the burden of encryption or decryption processes and then measuring if the program is given the burden of encryption and decryption processes. The initial execution of standard AES without encryption load, the amount of heap used is 11,613,416 bytes. Modified AES when given an encryption load of 10,950,208 bytes. The difference in execution time of the two programs reaches 663,208 bytes. In terms of program execution without adding the encryption process load from the two programs, it means that modified AES in terms of memory usage has more remaining space than standard AES.

The addition of the encryption process load from the five sample data files increases the heap size load. The average difference in the heap used on each standard and modified AES reaches 26,775,223 bytes. These results prove that the difference in heap size in modified AES can reduce memory allocation during encryption and without the burden of the encryption process, as shown in **TABLE 8**.

**TABLE 8.** Memory profiling of standard and modified AES decryption

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | File Name | Size | Heap used | |
| Standard | Modification |
|  | (.docx) | (Bytes) | (ms) | (ms) |
| 1 | 0 | 0 | 11.613.416 | 10.950.208 |
| 2 | 01A | 13.187 | 21.606.680 | 19.779.600 |
| 3 | 02A | 13.550 | 40.421.128 | 28.770.400 |
| 4 | 03A | 14.590 | 50.510.984 | 42.560.176 |
| 5 | 04A | 15.764 | 81.592.616 | 63.202.640 |
| 6 | 05A | 15.905 | 102.240.100 | 88.456.840 |
| Average | | | 61.596.984 | 34.821.761 |

The program execution without being given a load from both algorithms, each AES standard and AES modification, is 11,613,416 and 10,950,208. The change in heap size has an increase when given a decryption load of 5 files with different sizes, resulting in a heap size of each AES standard and AES modification of 67,136,860 bytes and 42,290,180 bytes. Based on tables 5 and 6, the average heap size when receiving encryption and decryption loads from both algorithms has a difference where the heap used tends to be higher during decryption compared to encryption.

The difference in the data results from the memory profiling test is due to improvisation in the mixcolumns transformation. Reducing the shift left operation can provide a memory reduction effect when the program gets an encryption and decryption load. Reducing the shift left operation does not have an impact on decreasing the percentage of the avalanche effect as in tables 1 and 2.

## PERFORMANCE ANALYSIS OF STANDARD AND MODIFIED AES

Testing the performance of an algorithm can use several variables such as time and memory. The amount of memory used can be measured based on memory profiling. The larger the memory allocation used, the lower the performance of an algorithm. The same is true for testing how much time a program uses to complete the data encryption and decryption process. The amount of computing time illustrates how much efficiency the time used in the algorithm and its program implementation is. The time and memory variables are considerations for how efficient standard AES and modified AES are.

Testing the performance of an algorithm can use several variables such as time and memory. The amount of memory used can be measured based on memory profiling. The larger the memory allocation used, the lower the performance of an algorithm. The same is true for testing how much time a program uses to complete the data encryption and decryption process. The amount of computing time illustrates how much efficiency the time used in the algorithm and its program implementation is. The time and memory variables are considerations for how efficient standard AES and modified AES are.

The computational speed between standard AES and modified AES has an average encryption time comparison of 18,905.8 ms and 656.6 ms. The average decryption time tends to be higher than the encryption process, which is 23,612 ms and 818 ms. The performance of both algorithms is measured based on the average heap used in the encryption execution of 5 files with a difference of 26 mb. The test results on the decryption process of the two algorithms produce an average heap size difference of 24 mb. The difference in data generated in the memory profiling test can be concluded that the heap used in the modified AES has been able to improve performance based on the memory allocation used during program execution.

Modified AES is proven to be able to reduce memory usage and encryption and decryption execution time based on test data on execution time and memory profiling. Test data shows that the algorithm performance increases as memory usage and time used to complete the encryption and decryption processes decrease.

# CONCLUSIONS

The performance of a cryptographic algorithm can be measured based on time and memory variables. AES improvisation with the aim of improving computing performance must consider power efficiency, cost, and data security. Based on research that has been conducted, the performance of modified AES has been proven to be able to improve data security, namely AE of 0.92%. Along with increasing data security, the performance of modified AES is able to reduce computing time and memory usage based on heap used.

Although there is an increase in confusion and diffusion in the modified AES, the increase in AE is still relatively close to the AE results of the standard AES. Therefore, in further research, it is necessary to develop the mixcolumn transformation with the main foundation of polynomial algebra GF(2^8) with modulo x^4+1 and its inverse. so that all changes in the mixcolumn constant matrix will be obtained from the inverse of the new polynomial equation.

# Acknowledgments

Thanks to the Faculty of Engineering who helped implement this Research program activity in the form of funding. We also thank the Informatics lab team who have supported the experiment so that this research has run smoothly.

# References

1. A. Aminudin and H. Hariyadi, “Analisa Kombinasi Algoritma Aes Dengan Blum-,” in *Sentra*, Malang: Universitas Muhammadiyah Malang, pp. 365–373, 2020.

2. H. M. Mohammad and A. A. Abdullah, “Enhancement process of AES: a lightweight cryptography algorithm-AES for constrained devices,” *Telkomnika (Telecommunication Comput. Electron. Control.*, vol. 20, no. 3, pp. 551–560, 2022.

3. A. Aminudin and E. Budi Cahyono, “A Practical Analysis of the Fermat Factorization and Pollard Rho Method for Factoring Integers,” *Lontar Komput. J. Ilm. Teknol. Inf.*, vol. 12, no. 1, p. 33, 2021.

4. S. William, *Cryptography and Network Security*. 2020.

5. R. Verma and A. K. Sharma, “Cryptography: Avalanche effect of AES and RSA,” *Int. J. Sci. Res. Publ.*, vol. 10, no. 4, p. p10013, 2020.

6. K. Muttaqin and J. Rahmadoni, “Analysis and Design of File Security System Aes (Advanced Encryption Standard) Cryptography Based,” *J. Appl. Eng. Technol. Sci.*, vol. 1, no. 2, pp. 113–123, 2020.

7. T. M. Kumar, K. S. Reddy, S. Rinaldi, B. D. Parameshachari, and K. Arunachalam, “A low area high speed fpga implementation of aes architecture for cryptography application,” *Electron.*, vol. 10, no. 16, 2021.