HARDWARE IMPLEMENTATION OF GALOIS FIELD MULTIPLICATION FOR MIXCOLUMN AND INVERSE MIXCOLUMN PROCESS IN ENCRYPTION-DECRIPTION ALGORITHMS

RAGIEL HADI PRAYITNO¹, SUNNY ARIEF SUDIRO²*, SARIFUDDIN MADEMDA³ AND SURYADI HARMANTO⁴

¹,²,³,⁴ Doctoral Program in Information Technology Gunadarma University, Jakarta – Indonesia
² Sekolah Tinggi Manajemen Informatika dan Komputer Jakarta STI&K, Jakarta – Indonesia
E-mail: ¹ragielhp@staff.gunadarma.ac.id, ²sunnyariefsudiro@ieee.org, ³sarif@staff.gunadarma.ac.id, ⁴misdie@staff.gunadarma.ac.id

ABSTRACT

This article described the Advanced Encryption Standard (AES) mixcolumns (MC) and inverse mixcolumns (IMC) process based on the multiplication of Galois Fields (GF 2⁸). Multiplied the original data and Matrix MC will have resulted an Encryption. In decryption, the IMC transformation method was same as which used during encryption, where the input for decryption was the encryption matrix array data and IMC matrix. The output which was generated in the decryption was the original matrix array data. The transformation of MC and IMC was conducted using two methods; they were based on the multiplication of Galois Field (GF2⁸) and based on tables L and E. The method in this article had been applied to MATLAB software and implemented in hardware using Field Programmable Gate Array (FPGA) device. In the implementation of hardware, it required 28 slice LUT, 28 LUT-FF and 24 bonded IOB with process time (delay) of 2.236 ns.

Keywords: AES, FPGA, Galois Field, Inverse Mixcolumns/Mixcolumns.

1. INTRODUCTION

Data security is one of the essentials to maintain the confidentiality of information. Data becomes more vulnerable to be breached because the data has been converted into digital form [1]. Digital storage and data media transmission data are vulnerable to attack from the hacker. If media containing data has been damaged, the data is vulnerable to be theft, which it can be misused. Therefore, data breach has potential to harm the certain parties. One of the methods that can be used to prevent its damage is cryptography method.

Cryptography is security method that has been existed for centuries. Cryptography method is implemented in communications from military to commercial. The development of internet and electronic commerce makes cryptography method as the important part of functioning global economy and has become something that millions of people use every day. Sensitive information such as bank records, credit card statements, passwords, or personal communications, must be encrypted, modified in order that it can be only accessed by authorized person or parties, and cannot be deciphered for the others [2].

Cryptography is a process of converting plain text message (or plain text) into encrypted text message (or ciphers) based on algorithm that is known as a sender and receiver, so that the encrypted message can be returned to its original form, it is plain text. Encrypted message cannot be read by anyone except the addressee, where the receiver has the same key as the sender. Encryption is the process of converting plain text message into cipher text form. Decryption is the process of converting ciphertext into plain text message [3].

Each encryption and decryption process has a key, where the key can be a word or phrase. The key is a part of cryptography method which used for securing the data. AES is encryption which adopted as a Federal Information Processing Standards (FIPS) by National Institute of Standards and Technology (NIST) in 2001 [4]. AES algorithm has several processes, they are: Addroundkey, performed bitwise xor of round
key. Subbytes, performed substitution matrix array data using S-Box table. Shiftrows, changed matrix array data. Mixcolumns, performed randomization matrix array data.

Mix Columns transformation (MC) is part of encryption process in AES. The Inverse Mix Columns (IMC) transformation is part of decryption process in AES algorithm. The transformation can be used for securing data by randomizing the original data in state array column, thus it can obtain new data in each state array column. MC/IMC transformation consumed more power and logic thus the optimization is important [5]. In traditional AES, MC and IMC are implemented as separate modules, with exception of some implementation which used sharing resource between MC/IMC to optimize power and space [6, 7]. The design which was proposed in previous studies was based on bytes, decomposition matrix and minimization equation using common sub-expression elimination [6].

Architecture of a low-power FPGA is based on using AES Mix Columns / Inverse Mix Columns transformation. The implementation used decomposing techniques, pre-computing, and powering parallel to reduce a power circuit. The experimental result showed that circuit consumption was reduced compared to the previous implementation. The reduce circuit power in the average of 28% was achieved [5]. Instead of using two different modules of mixed column transformation and inversed, one module can be applied for both transformations. It reduced the entire consumption area of AES algorithm [9].

The previous researcher used memory-less combinatorial design for SB/ISR implementation as an alternative to achieve a higher-speeds by eliminating memory access delay while maintaining or increasing the efficiency of used the entire component area. This study is also explored the use of sub-pipelining to more increase the speed and throughput of the suggestion implementation. FPGA architecture used optimization in both of inverter design and isomorphic mapping using composite field arithmetic to reduce the area requirements of the used components [10].

The transmission of information through transmission media is insecure and has the potential to be accessed by unauthorized parties. The security method that can be applied is to perform hardware-based information scrambling (encryption/decryption) bit by bit.

This research is about designing Galois Field Multiplication process component to be used in encryption/decryption process rather than using look table (Table L and E) which will cause the use of many resources and large delays (latency).

In this paper, we focus on the design of MC/IMC transformation in AES based on Galois field multiplication $2^8$ (GF$2^8$) in hardware based environment to obtain a device/component with faster of process comparing to other study that in software environment is slower. This design is suitable for real time application with minimal resources of FPGA devices. The experiment was conducted on hardware Intel Core i7 Generation 8, 8GB ram using Matlab and Xilinx ISE Software.

2. LITERATURE REVIEW

Cryptography method with mixcolumns transformation can be developed in software and hardware. Therefore, some mixcolumns transformation methods are analyzed in this study to find easy-to-follow and safe method by software and suitable for hardware implementation.

2.1 Mixcolumn Transformation

MixColumns transformation, the column in state (which is a word) is treated as a polynomial with the coefficient in GF($2^8$).

\[ a_3, a_2, a_1, a_0 \]

Word in formula (1) is treated as a polynomial, thus it becomes as follows:

\[ a_3x^3 + a_2x^2 + a_1x + a_0 \] (2)

The addition of a polynomial is conducted because it is added to polynomial ring. In order that, the result remains as big as a word, the multiplication is performed modulo polynomial

\[ g(x) = x^4 + 1 \] (3)

\[ g(x) = x^4 + 1 \] is the equation of decimal value 17, where X^4 represents the value $2^4$ and 1 represents $2^0$. It is easy to indicate that

\[ X^4 \mod (X^4 + 1) = X^4 \mod 4 \] (4)

because -1 = 1 in GF($2^8$). Since X^4 + 1 is not irreducible polynomial in $K[X]$ where $K = GF(2^8)$, then $K[X]/g(X)K[X]$ is not a field: not all polynomial has inverse. However, the polynomial
\[ a(x) = 03X^3 + 01X^2 + 01X + 02 \] (5)

has inverse
\[ a^{-1}(x) = 0Bx^3 + 0DX^2 + 09X + 0E \] (6)

The values of formulas (5) and (6) are hexadecimal numbers (0B=11, 0D=13 and 0E=14). The MixColumns transform multiplies (modulo polynomial g(X)) each column in the state (treated as a polynomial) by the polynomial a(X). The MixColumns transformation of the state can be formulated for its effect in each column c as follows [4]:

\[
\begin{bmatrix}
S'_{0,c} \\
S'_{1,c} \\
S'_{2,c} \\
S'_{3,c}
\end{bmatrix} =
\begin{bmatrix}
02 & 03 & 01 & 01 \\
01 & 02 & 03 & 01 \\
01 & 01 & 02 & 03 \\
03 & 01 & 01 & 03
\end{bmatrix}
\begin{bmatrix}
S_{0,c} \\
S_{1,c} \\
S_{2,c} \\
S_{3,c}
\end{bmatrix}
\] (7)

\(S_{0,c}, S_{1,c}, S_{2,c}, S_{3,c}\) are the original data. \(S'_{0,c}\) is data change in the first row of column c which resulted from MC/IMC transformation process between MC matrix and original data. Afterward for \(S'_{1,c}, S'_{2,c}, \text{ and } S'_{3,c}\) are data change in row 2 to row 4 column c.

MC/IMC operations are used in Advanced Encryption Standard for the purpose of hardware implementation in restricted environments. This study was supported by mathematical analysis of both transformations and caused serial decomposition and efficient parallel [7].

The previous research created a high-efficient architecture to perform mixed column operations, which is the main function in AES method. This study used prehistoric Vedic Mathematics techniques which can provide more efficient results in AES [8].

2.2 Mean Square Error (MSE)

Mean squared error (MSE) is the most widely used and simplest reference metric that is calculated by the difference in the intensity of the squares of the distorted and reference values. MSE is the most common quality measurement metric estimator, the squaring part of the function magnifies the error [11], see eq.8.

\[ MSE = \frac{1}{m} \sum_{j=1}^{m} (X_j - Y_j)^2 \] (8)

(best value = 0; worst value = +∞, X is first value (as reference) and Y is second value to be compare)

3. PROPOSED METHOD

The method was developed referred to the calculation of polynomial function GF(2^8). The research was conducted using MATLAB software. Encryption input was in the form of a multiplication data and Matrix MixColumns. The output encryption is in the form of encryption data. Figure 1 illustrated the MixColumns transformation encryption process was used in this study. The first process was the multiplication of Galois Field between the data matrix and the MC matrix, if the multiplication result exceeded 255, it was necessary to do irreducible to that value. The irreducible process needed to be applied to values of \(x^8\) and above. The next process was to XOR the multiplication result value to get a new matrix value in the MC transformation. In decryption, IMC transformation method was the same as that used during encryption, where the input to decryption was encryption matrix array data and IMC matrix. The output that was generated in decryption was the original matrix array data.

![Figure 1: MC/IMC Transformation](image-url)
3.1 Data

The data applied in this study was digital data in the form of 16 decimal numbers that was divided into 4x4 matrixes, for example:

\[
\begin{bmatrix}
170 & 249 & 69 & 159 \\
249 & 80 & 183 & 251 \\
143 & 80 & 80 & 239 \\
60 & 239 & 239 & 64
\end{bmatrix}
\]

3.2 Matrix MC/IMC

MC/IMC is treated as a polynomial with coefficients in GF2^8, based on (5) data matrix MC and data matrix IMC, based on (6).

\[
Mix = \begin{bmatrix}
2 & 3 & 1 & 1 \\
1 & 2 & 3 & 1 \\
1 & 1 & 2 & 3 \\
3 & 1 & 1 & 2
\end{bmatrix}
\]

\[
InvMix = \begin{bmatrix}
14 & 11 & 13 & 9 \\
9 & 14 & 11 & 13 \\
13 & 9 & 14 & 11 \\
11 & 13 & 9 & 14
\end{bmatrix}
\]

3.3 Galois Field Multiplication

Galois field multiplication had been conducted to randomize the original array data. Multiplication process had been conducted as follows (see Figure 2):

1. The first column of matrix array data had been multiplied by the first row of matrix MC, the result which had obtained is used for restoring matrix array data in row 1 column 1 (1,1).
2. The first column of matrix array data had been multiplied by the second row of matrix MC, the results which had obtained is used for restoring matrix array data in row 2 column 1 (2,1).
3. The first column of matrix array data had been multiplied by the third row of matrix MC, the results which had obtained is used for restoring matrix array data in row 3 column 1 (3,1).
4. The first column of matrix array data had been multiplied by the fourth row of matrix MC, the results which had obtained is used for restoring matrix array data in row 4 column 1 (4,1).

5. The next step, repeated the steps 1 to 4 for the next column matrix array data.

\[
\begin{bmatrix}
170 & 249 & 69 & 159 \\
249 & 80 & 183 & 251 \\
143 & 80 & 80 & 239 \\
60 & 239 & 239 & 64
\end{bmatrix}
\times
\begin{bmatrix}
2 & 3 & 1 & 1 \\
1 & 2 & 3 & 1 \\
1 & 1 & 2 & 3 \\
3 & 1 & 1 & 2
\end{bmatrix}
\]

\[
\text{Figure 2: Galois field multiplication}
\]

For example, index (I) data(1,1) had been multiplied by index MC(1,1):

Index (I) data(1,1) = 170 convert to binary
\[= 10101010_2\]
\[= 2^7 + 2^5 + 2^3 + 2^1\]

Index (I) MC(1,1) = 2 converted to binary
\[= 00000010_2\]
\[= 2^1\]

I data(1,1) * I MC(1,1) = \((2^7 + 2^5 + 2^3 + 2^1)(2^1)\)
\[= 2^8 + 2^6 + 2^4 + 2^2\]
\[= 101010100_2\]
\[= 340\]

3.4 Irreducible Polynomial

Based on the example of the previous subsection Galois field multiplication, irreducible needed to be applied on \(x^8\), \(x^9\), and etc. In the previous discussion, the multiplication result is \(2^8 + 2^6 + 2^4 + 2^2\), so the result was irreducible.

\[
= 2^8 + 2^6 + 2^4 + 2^2 \\
= (2^4 + 2^2 + 2 + 1) + (2^4 + 2^2) \\
= 2^6 + 2^3 + 2^2 + 2 + 1 \\
= 01001111_2 \\
= 79
\]

Irreducible result of data index(1,1) and MC index(1,1) was 79 in decimal form. The results were XORed by multiplying: data index(2,1) with MC index(1,2), index data(3,1) with MC index(1,3), and data index(4,1) with MC index(1,4).
4. RESULT AND DISCUSSION

The first research experiment had been conducted using MATLAB software. In this experiment, two different methods were used:
1. Based on multiplication of Galois field (GF2\(^8\)).
2. Based on the tables L and E [12].
3. Implementation using FPGA.

4.1 Experiment Based on Multiplication of Galois Field (GF2\(^8\))

Matrix data and matrix MC had been multiplied using Galois field multiplication algorithm to produce a change to the previous matrix data. The result of matrix data changes had been multiplied by matrix IMC using Galois field to return the matrix data to its original.

\[
\begin{bmatrix}
170 & 249 & 69 & 159 \\
249 & 80 & 183 & 251 \\
143 & 80 & 239 & 60 \\
60 & 239 & 239 & 64
\end{bmatrix}
\]

\[
\begin{bmatrix}
2 & 3 & 1 & 1 \\
1 & 2 & 3 & 1 \\
1 & 1 & 2 & 3 \\
3 & 1 & 1 & 2
\end{bmatrix}
\]

MC/IMC transformation in the encryption/decryption process was as followed:

\[
\begin{align*}
\text{Index}(1,1) &= (\text{data}(1,1) \ast \text{mix}(1,1)) \oplus (\text{data}(2,1) \ast \text{mix}(1,2)) \oplus (\text{data}(3,1) \ast \text{mix}(1,3)) \oplus (\text{data}(4,1) \ast \text{mix}(1,4)) \\
\text{Index}(1,2) &= (\text{data}(1,1) \ast \text{mix}(2,1)) \oplus (\text{data}(2,1) \ast \text{mix}(2,2)) \oplus (\text{data}(3,1) \ast \text{mix}(2,3)) \oplus (\text{data}(4,1) \ast \text{mix}(2,4)) \\
\text{Index}(1,3) &= (\text{data}(1,1) \ast \text{mix}(3,1)) \oplus (\text{data}(2,1) \ast \text{mix}(3,2)) \oplus (\text{data}(3,1) \ast \text{mix}(3,3)) \oplus (\text{data}(4,1) \ast \text{mix}(3,4)) \\
\text{Index}(1,4) &= (\text{data}(1,1) \ast \text{mix}(4,1)) \oplus (\text{data}(2,1) \ast \text{mix}(4,2)) \oplus (\text{data}(3,1) \ast \text{mix}(4,3)) \oplus (\text{data}(4,1) \ast \text{mix}(4,4)) \\
\end{align*}
\]

The Galois field multiplication (GF2\(^8\)) above had been conducted to produce changed to the previous matrix data and returned the matrix data to its original. The multiplication for Index (1,1) showed that data had been be changed was row 1 column 1 of matrix data and for the other index multiplication which showed the changed row and column. Symbol * was the multiplication between index data and index mix, so a symbol was XOR process after the multiplication of each index was conducted.

Based on experiment it was conducted with Matlab software, Figure 3(A) showed the time required to perform the MC transformation was 0.005 seconds, where the dec2bin function took 0.004 seconds and the ujicoba4_mix function took 0.001 seconds. Figure3(B) showed the IMC transformation process took 0.006 seconds, where dec2bin function took 0.001 seconds, dec2bin function took 0.003 seconds, gfmul function took 0.001 seconds, and the ujicoba3 function took 0.001 seconds.

![Figure 3: Galois Field Multiplication Encryption (A) and Decryption (B) Time Process](image-url)
4.2 Experiment Based on Lookup Table

In the second experiment, the data that had been used in accordance with Figure 2. Figure 4(A) was a look up table called table L and E for MC/IMC. Matrix data and matrix MC were multiplied based on table L and table E to change the previous matrix data. The result of matrix data changed had been multiplied by matrix IMC based on table L and table E to return the matrix data to its original. The multiplication results based on table L and table E were search results with table L then followed by summing the results, followed by searching with table E. The search results with tables L and E were only performed on matrix index MC with values 2 and 3. MC transformation in encryption process was as followed:

\[
\text{Index} (1,1) = (\text{data}(1,1) \times \text{mix}(1,1)) \oplus (\text{data}(2,1) \times \text{mix}(1,2)) \oplus (\text{data}(3,1) \times \text{mix}(1,3)) \oplus (\text{data}(4,1) \times \text{mix}(1,4))
\]

\[
= E(L(\text{data}(1,1)) + L(\text{mix}(1,1))) \oplus E(L(\text{data}(2,1)) + L(\text{mix}(1,2))) \oplus E(\text{data}(3,1)) \oplus \text{data}(1,1)
\]

In IMC transformation process, the multiplication based on table L and table E was the result with table L then followed by the sum of the results, followed by a search with table E. IMC transformation in the decryption process was as followed:

\[
\text{Index} (1,1) = (\text{data}(1,1) \times \text{mix}(1,1)) \oplus (\text{data}(2,1) \times \text{mix}(1,2)) \oplus (\text{data}(3,1) \times \text{mix}(1,3)) \oplus (\text{data}(4,1) \times \text{mix}(1,4))
\]

\[
= E(L(\text{data}(1,1)) + L(\text{mix}(1,1))) \oplus E(L(\text{data}(2,1)) + L(\text{mix}(1,2))) \oplus E(\text{data}(3,1)) \oplus (\text{data}(3,1) \times \text{mix}(1,3))
\]

Multiplication based on table L and table E (Figure 4) had been conducted to produce changed to the previous matrix data and returned the matrix data to its original. The multiplication for Index (1,1) showed that the data changed was row 1 column 1 of matrix data and for the other index multiplication which indicate changed the row and column. Based on experiments which was conducted with Matlab software, Figure 5 indicated the time required to perform MC transformation was 0.006 seconds and indicated the IMC transformation process took 0.007 seconds.
4.3 Implementation using FPGA

Based on behavioral algorithm in Matlab, we proposed the implementation of Galois field multiplication in hardware environment using FPGA device and ISE Software for Xilinx. Figure 6 was a concept of implementing the MixColumns transformation with VHDL programming based on behavioral algorithms in Matlab.

The input data in the transformation was declared as “data” with type std_logic_vector(7 downto 0) or 8 bit data. The output of the transformation was declared as “douts” with type std_logic_vector(7 downto 0) or 8 bit output data. This transformation used 3 memories consisting of 1 constant memory and 2 memories as temporary storage of the resulting data. Constant memory was used for MixColumns data of 16 memory cells of type std_logic_vector(7 downto 0). The second memory was declared as mem1 with type std_logic_vector(7 downto 0) on ram 1 of 16 memory cells. The third memory was then declared as mem2 with type std_logic_vector(7 downto 0) of 16 memory cells which were used to store the results to be output to “douts”. The Galois field multiplication was conducted in the MixColumns transformation. The multiplication result was processed in memory to produce a new matrix data of MixColumns transformation which was stored in mem2. The pseudo code of Galois Multiplication in hardware implementation were:

```vhdl
entity Galois01 is
    Port ( a : in  STD_LOGIC_VECTOR (7 downto 0);
            b : in  STD_LOGIC_VECTOR (7 downto 0);
            c : out STD_LOGIC_VECTOR (7 downto 0));
end Galois01;

architecture Behavioral of Galois01 is
begin
    process(a,b)
    variable bobot : integer range 0 to 15;
    variable hasil : std_logic_vector(16 downto 0);
    begin
        hasil := (others=>'0');
        for i in 0 to 7 loop
            for j in 0 to 7 loop
                if (a(i) and b(j)) ='0' then
                    next;
                else
                    bobot:= i+j;
                    if hasil(bobot)='1' then
                        hasil(bobot):='0';
                    else
                        hasil(bobot):='1';
                        end if;
                    end if;
                end loop;
            end loop;
        if hasil(11)='1' then
            hasil(7 downto 0):=hasil(7 downto 0) xor "11011000";
            end if;
        if hasil(10)='1' then
            hasil(7 downto 0):=hasil(7 downto 0) xor "01101100";
            end if;
        if hasil(9)='1' then
            hasil(7 downto 0):=hasil(7 downto 0) xor "00110110";
            end if;
        if hasil(8)='1' then
            hasil(7 downto 0):=hasil(7 downto 0) xor "00011011";
            c<= hasil(7 downto 0);
            end process;
            end Behavioral;
end Galois01;
```

Figure 6: Concept Of Implementing The MixColumns Transformation

The resource that had been used for implementing Galois field multiplication in environment with ISE software for Xilinx based on behavioral algorithms in Matlab were 28 slices LUTs, 28 LUT-FF and 24 IOB bonded with...
process time (delay) of 2.236 ns. Figure 6(A) showed the simulation results with ISIM simulator. The simulation results of Figure 7(A) indicated that Galois calculation for 129 and 13 was 215, 102 and 11 was 183, 4 and 13 was 56, 229 and 9 was 140. All experimental results in Figure 7(A) were same as those in Matlab. All experiments result for 3 methods were presented in Table 1. Figure 7(B) showed resources summary for the design and device utilization.

### Table 1: Comparison of method 1 and method 2 with proposed method with respect to MSE

<table>
<thead>
<tr>
<th>No</th>
<th>Data1</th>
<th>Data2</th>
<th>Method 1</th>
<th>Method 2</th>
<th>Proposed Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>129</td>
<td>13</td>
<td>215</td>
<td>215</td>
<td>215</td>
</tr>
<tr>
<td>2</td>
<td>102</td>
<td>11</td>
<td>183</td>
<td>183</td>
<td>183</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>14</td>
<td>56</td>
<td>56</td>
<td>56</td>
</tr>
<tr>
<td>4</td>
<td>229</td>
<td>9</td>
<td>140</td>
<td>140</td>
<td>140</td>
</tr>
<tr>
<td>5</td>
<td>170</td>
<td>2</td>
<td>79</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>6</td>
<td>249</td>
<td>3</td>
<td>16</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>236</td>
<td>14</td>
<td>127</td>
<td>127</td>
<td>127</td>
</tr>
<tr>
<td>8</td>
<td>245</td>
<td>11</td>
<td>237</td>
<td>237</td>
<td>237</td>
</tr>
<tr>
<td>9</td>
<td>18</td>
<td>13</td>
<td>202</td>
<td>202</td>
<td>202</td>
</tr>
<tr>
<td>10</td>
<td>235</td>
<td>9</td>
<td>242</td>
<td>242</td>
<td>242</td>
</tr>
</tbody>
</table>

![Image](A)

**Figure 7**: Galois Field Multiplication Simulation Results With ISIM Simulator For 10 Experiments (A) And Galois Field Multiplication Device Utilization (B)

Based on the results obtained in table 1, the MSE result value between Exp. Method 3 (based on hardware implementation using FPGA device) compare to Exp. Method 1 and 2 (in Matlab programming, or software platform) for 10 data experiment are 0, there no difference result from hardware implementation and software platform, see in detail below:

\[
MSE = \frac{1}{10} \left( (215-215)^2 + (183-183)^2 + \ldots + (242-242)^2 \right) = 0
\]

The proposed $GF_{2^8}$ multiplication has been implemented by involving the resources of 28 LUTs with a processing time of 2.236 ns. This $GF_{2^8}$ multiplication is really used in a full MC/IMC circuit in which all processes occupy a resource of 214 LUTs and have a processing time of 2.7 ns [13]. Table 2 shows a comparison of the performance of the proposed method against the other four methods. The processing time of the proposed method is faster than the methods in [6, 7, 9], and the LUTs resources used are less than the method [5].
Table 2: Comparison Of MC/IMC With Proposed Method With Respect To Area And Processing Time.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No Of LUTs</td>
<td>544</td>
<td>43</td>
<td>92</td>
<td>118</td>
<td>28/214</td>
</tr>
<tr>
<td>2</td>
<td>Processing Time (ns)</td>
<td>1.6</td>
<td>5.9</td>
<td>3.68</td>
<td>7.16</td>
<td>2.236/2.7</td>
</tr>
</tbody>
</table>

4. CONCLUSIONS
MC/IMC transformation using tables L and E were slower than GF2$^8$ multiplication because, in the look-up tables’ process, there were two conversions employing tables L and E. The MSE generated between the experimental methods 1, 2 and 3 which have been carried out for 10 experiments was 0.

In hardware implementation using FPGA device, a component description (IP core) of GF2$^8$ Multiplication process is obtained with 28 slices LUT, 28 LUT-FF, 24 bonded IOB and process time (delay) of 2.236 ns, faster than in Matlab.

5. ACKNOWLEDGMENT
This work is supported by Research Program Kemendikbudristek Indonesia in Hibah Penelitian Disertasi Contract Number : 064/SP2H/LT/DRPM/2021 dated 18 March 2021 and Yayasan Pendidikan Gunadarma Jakarta Indonesia, Contract Number : 05A.26/LP/UG/IV/2021, 5 April 2021.

REFERENCES
