Journal of Theoretical and Applied Information Technology
31st July 2022. Vol.100. No 14
© 2022 Little Lion Scientific

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

HARDWARE IMPLEMENTATION OF GALOIS FIELD MULTIPLICATION FOR MIXCOLUMN AND INVERSEMIXCOLUMN PROCESS IN ENCRYPTION-DECRYPTION ALGORITHMS

RAGIEL HADI PRAYITNO[1], SUNNY ARIEF SUDIRO[2*], SARIFUDDIN MADENDA[3] AND SURYADI HARMANTO[4]

[1,3,4] Doctoral Program in Information Technology Gunadarma University, Jakarta – Indonesia

[2] Sekolah Tinggi Manajemen Informatika dan Komputer Jakarta STI&K, Jakarta – Indonesia

E-mail: [1]ragielhp@staff.gunadarma.ac.id, [2]sunnyariefsudiro@ieee.org, [3]sarif@staff.gunadarma.ac.id, [4]misdie@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 2^8). 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 (GF2^8) 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 (GF2^8) 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).

$$[a3,a2,a1,a0] \quad (1)$$

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

$$a3X^3 + a3X^2 + a3X^1 + a0 \quad (2)$$

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

$$g(x)=X^4 + 1 \quad (3)$$

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

$$X^i \bmod (X^4 + 1) = X^{i \bmod 4} \quad (4)$$

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

$$a(x) = 03X^3 + 01X^2 + 01X + 02 \quad (5)$$

has inverse

$$a^{-1}(x) = 0BX^3 + 0DX^2 + 09X + 0E \quad (6)$$

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]:

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad (7)$$

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

$$MSE = \frac{1}{m}\sum_{i=1}^{m}(X_i - Y_i)^2 \quad (8)$$

(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.

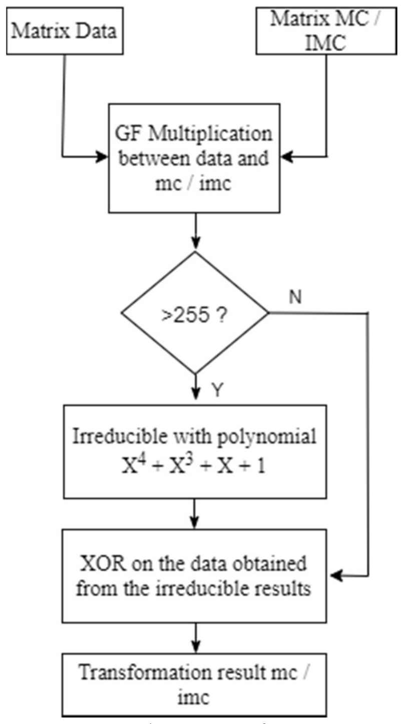

Matrix Data
Matrix MC / IMC
GF Multiplication between data and mc / imc
>255 ?
N
Y
Irreducible with polynomial X⁴ + X³ + X + 1
XOR on the data obtained from the irreducible results
Transformation result mc / imc

Figure 1: MC/IMC Transformation

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).

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

$$\text{Inv}Mix = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix}$$

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.

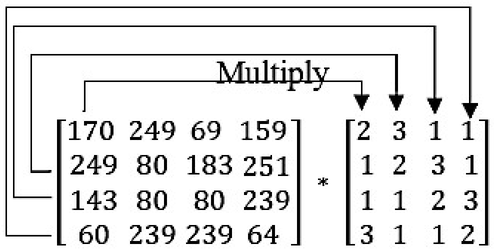


Figure 2: Galois field multiplication

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) = $(2^7+2^5+2^3+2^1)*(2^1)$

$= 2^{7+1} + 2^{5+1} + 2^{3+1} + 2^{1+1}$

$= 2^8+2^6+2^4+2^2$

$=101010100^2$

$= 340$

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.

$= 2^8 +2^6 +2^4 +2^2$
$= (2^4+2^3+2^1+1)+2^6+2^4+2^2$
$= 2^6 + 2^3 + 2^2 + 2^1 + 1$
$= 01001111_2$
$= 79$

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).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

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.

$$Data = \begin{bmatrix} 170 & 249 & 69 & 159 \\ 249 & 80 & 183 & 251 \\ 143 & 80 & 80 & 239 \\ 60 & 239 & 239 & 64 \end{bmatrix} \quad Mix = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

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

Index(1,1) = (data(1,1) * mix(1,1)) ⊕ (data(2,1) * mix(1,2)) ⊕ (data(3,1) * mix(1,3)) ⊕ (data(4,1) * mix(1,4))
Index(2,1) = (data(1,1) * mix(2,1)) ⊕ (data(2,1) * mix(2,2)) ⊕ (data(3,1) * mix(2,3)) ⊕ (data(4,1) * mix(2,4))
Index(3,1) = (data(1,1) * mix(3,1)) ⊕ (data(2,1) * mix(3,2)) ⊕ (data(3,1) * mix(3,3)) ⊕ (data(4,1) * mix(3,4))
Index(4,1) = (data(1,1) * mix(4,1)) ⊕ (data(2,1) * mix(4,2)) ⊕ (data(3,1) * mix(4,3)) ⊕ (data(4,1) * mix(4,4))

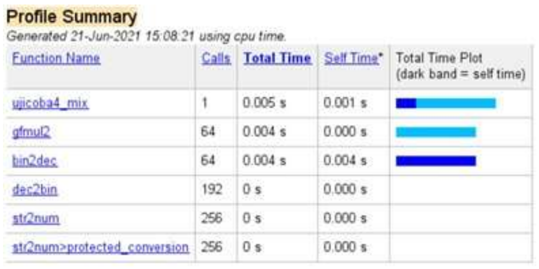
Index(1,2) = (data(1,2) * mix(1,1)) ⊕ (data(2,2) * mix(1,2)) ⊕ (data(3,2) * mix(1,3)) ⊕ (data(4,2) * mix(1,4))
Index(2,2) = (data(1,2) * mix(2,1)) ⊕ (data(2,2) * mix(2,2)) ⊕ (data(3,2) * mix(2,3)) ⊕ (data(4,2) * mix(2,4))
Index(3,2) = (data(1,2) * mix(3,1)) ⊕ (data(2,2) * mix(3,2)) ⊕ (data(3,2) * mix(3,3)) ⊕ (data(4,2) * mix(3,4))
Index(4,2) = (data(1,2) * mix(4,1)) ⊕ (data(2,2) * mix(4,2)) ⊕ (data(3,2) * mix(4,3)) ⊕ (data(4,2) * mix(4,4))

Index(1,3) = (data(1,3) * mix(1,1)) ⊕ (data(2,3) * mix(1,2)) ⊕ (data(3,3) * mix(1,3)) ⊕ (data(4,3) * mix(1,4))
Index(2,3) = (data(1,3) * mix(2,1)) ⊕ (data(2,3) * mix(2,2)) ⊕ (data(3,3) * mix(2,3)) ⊕ (data(4,3) * mix(2,4))
Index(3,3) = (data(1,3) * mix(3,1)) ⊕ (data(2,3) * mix(3,2)) ⊕ (data(3,3) * mix(3,3)) ⊕ (data(4,3) * mix(3,4))
Index(4,3) = (data(1,3) * mix(4,1)) ⊕ (data(2,3) * mix(4,2)) ⊕ (data(3,3) * mix(4,3)) ⊕ (data(4,3) * mix(4,4))

Index(1,4) = (data(1,4) * mix(1,1)) ⊕ (data(2,4) * mix(1,2)) ⊕ (data(3,4) * mix(1,3)) ⊕ (data(4,4) * mix(1,4))
Index(2,4) = (data(1,4) * mix(2,1)) ⊕ (data(2,4) * mix(2,2)) ⊕ (data(3,4) * mix(2,3)) ⊕ (data(4,4) * mix(2,4))
Index(3,4) = (data(1,4) * mix(3,1)) ⊕ (data(2,4) * mix(3,2)) ⊕ (data(3,4) * mix(3,3)) ⊕ (data(4,4) * mix(3,4))
Index(4,4) = (data(1,4) * mix(4,1)) ⊕ (data(2,4) * mix(4,2)) ⊕ (data(3,4) * mix(4,3)) ⊕ (data(4,4) * mix(4,4))

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.

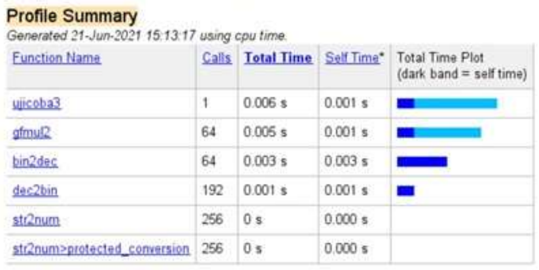
Based on experiment it was conducted with Matlab software, Figure 3(A) showed the time required to perform the MC transformation was 0.005 seconds, where the dec2bin function took 0.004 seconds and the ujicoba4_mix function took 0.001 seconds. Figure3(B) showed the IMC transformation process took 0.006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



Profile Summary
Generated 21-Jun-2021 15:08:21 using cpu time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
ujicoba4_mix	1	0.005 s	0.001 s	
gfmul2	64	0.004 s	0.000 s	
bin2dec	64	0.004 s	0.004 s	
dec2bin	192	0 s	0.000 s	
str2num	256	0 s	0.000 s	
str2num>protected_conversion	256	0 s	0.000 s	

(A)



Profile Summary
Generated 21-Jun-2021 15:13:17 using cpu time.

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
ujicoba3	1	0.006 s	0.001 s	
gfmul2	64	0.005 s	0.001 s	
bin2dec	64	0.003 s	0.003 s	
dec2bin	192	0.001 s	0.001 s	
str2num	256	0 s	0.000 s	
str2num>protected_conversion	256	0 s	0.000 s	

(B)

Figure 3: Galois Field Multiplication Encryption (A) and Decryption (B) Time Process

4.2 Experiment Based on Lookup Table

In second experiment, the data that had been used in accordance with Figure2. 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:

$$\begin{aligned}\text{Index }(1,1) &= (data(1,1)*mix(1,1)) \oplus (data(2,1)*mix(1,2)) \oplus (data(3,1)*mix(1,3)) \oplus (data(4,1)*mix(1,4)) \\ &= E(L(data(1,1)) + L(mix(1,1))) \oplus E(L(data(2,1)) + L(mix(1,2))) \oplus data(3,1) \oplus data(4,1)\end{aligned}$$

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:

$$\begin{aligned}\text{Index }(1,1) &= (data(1,1)*mix(1,1)) \oplus (data(2,1)*mix(1,2)) \oplus (data(3,1)*mix(1,3)) \oplus (data(4,1)*mix(1,4)) \\ &= E(L(data(1,1)) + L(mix(1,1))) \oplus E(L(data(2,1)) + L(mix(1,2))) \oplus E(L(data(3,1)) + L(mix(1,3))) \\ &\quad \oplus E(L(data(4,1)) + L(mix(1,4)))\end{aligned}$$

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.

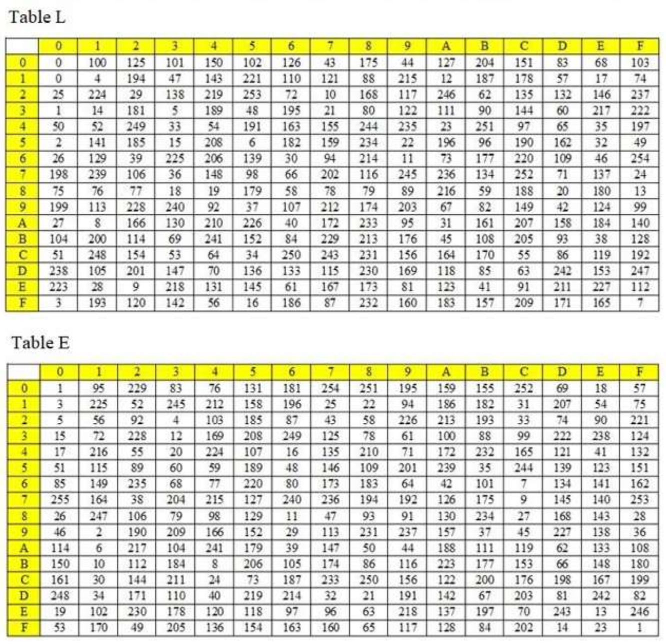


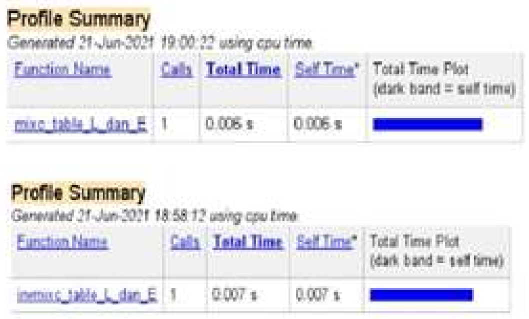
Table L

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	100	125	101	150	102	126	43	175	44	127	204	151	83	68	103
1	0	4	194	47	143	221	110	121	88	215	12	187	178	57	17	74
2	25	224	29	138	219	253	72	10	168	117	246	62	135	132	146	237
3	1	14	181	5	189	48	195	21	80	122	111	90	144	60	217	222
4	50	52	249	33	54	191	163	155	244	235	23	251	97	65	35	197
5	2	141	185	15	208	6	182	159	234	22	196	96	190	162	32	49
6	26	129	39	225	206	139	30	94	214	11	73	177	220	109	46	254
7	198	239	106	36	148	98	66	202	116	245	236	134	252	71	137	24
8	75	76	77	18	19	179	58	78	79	89	216	59	188	20	180	13
9	199	113	228	240	92	37	107	212	174	203	67	82	149	42	124	99
A	27	8	166	130	210	226	40	172	233	95	31	161	207	158	184	140
B	104	200	114	69	241	152	84	229	213	176	45	108	205	93	38	128
C	51	248	154	53	64	34	250	243	231	156	164	170	55	86	119	192
D	238	105	201	147	70	136	133	115	230	169	118	85	63	242	153	247
E	223	28	9	218	131	145	61	167	173	81	123	41	91	211	227	112
F	3	193	120	142	56	16	186	87	232	160	183	157	209	171	165	7

Table E

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	1	95	229	83	76	131	181	254	251	195	159	155	252	69	18	57
1	3	225	52	245	212	158	196	25	22	94	186	182	31	207	54	75
2	5	56	92	4	103	185	87	43	58	226	213	193	33	74	90	221
3	15	72	228	12	169	208	249	125	78	61	100	88	99	222	238	124
4	17	216	55	20	224	107	16	135	210	71	172	232	165	121	41	132
5	51	115	89	60	59	189	48	146	109	201	239	35	244	139	123	151
6	85	149	235	68	77	220	80	173	183	64	42	101	7	134	141	162
7	255	164	38	204	215	127	240	236	194	192	126	175	9	145	140	253
8	26	247	106	79	98	129	11	47	93	91	130	234	27	168	143	28
9	46	2	190	209	166	152	29	113	231	237	157	37	45	227	138	36
A	114	6	217	104	241	179	39	147	50	44	188	111	119	62	133	108
B	150	10	112	184	8	206	105	174	86	116	223	177	153	66	148	180
C	161	30	144	211	24	73	187	233	250	156	122	200	176	198	167	199
D	248	34	171	110	40	219	214	32	21	191	142	67	203	81	242	82
E	19	102	230	178	120	118	97	96	63	218	137	197	70	243	13	246
F	53	170	49	205	136	154	163	160	65	117	128	84	202	14	23	1

Figure 4: Table L and Table E [12]


Profile Summary
Generated 21-Jun-2021 19:00:22 using cpu time

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
mixc_table_L_dan_E	1	0.006 s	0.006 s	▬

Profile Summary
Generated 21-Jun-2021 18:58:12 using cpu time

Function Name	Calls	Total Time	Self Time*	Total Time Plot (dark band = self time)
inemixc_table_L_dan_E	1	0.007 s	0.007 s	▬

Figure 5: Encryption and Decryption Time Processing based on table L and E

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 :

```
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');   -- 1 + x^2 + x^3 + x^4 +
x^8
 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"; -- "11011000" = 216 = 8*27
 end if;
 if hasil(10)='1' then
    hasil(7 downto 0):=hasil(7 downto 0) xor
"01101100"; -- "01101100" = 108 = 4*27
 end if;
 if hasil(9)='1' then
    hasil(7 downto 0):=hasil(7 downto 0) xor
"00110110"; -- "00110110" = 54 = 2*27
end if;
if hasil(8)='1' then
   hasil(7 downto 0):=hasil(7 downto 0) xor
"00011011";--"00011011" = 27
end if;
c<= hasil(7 downto 0);
 end process ;
 end Behavioral;
```

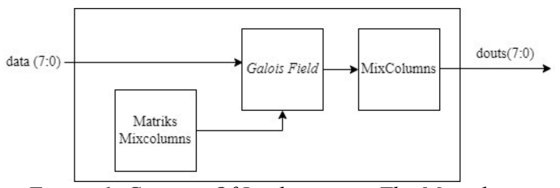
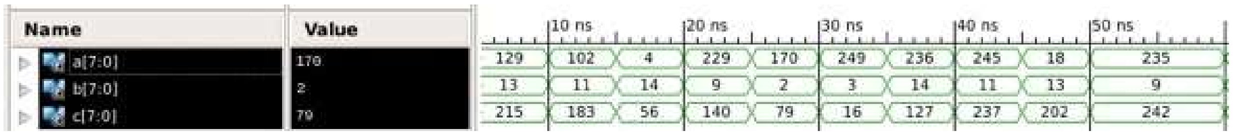


Figure 6 :Concept Of Implementing The Mixcolumns Transformation

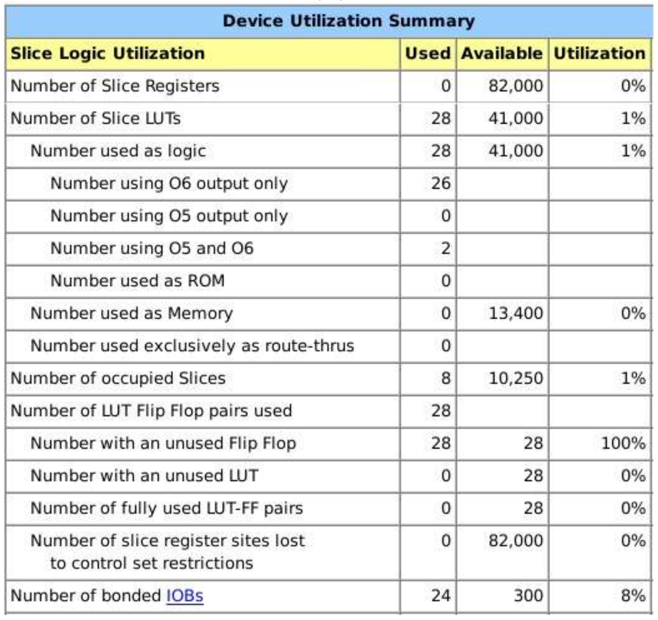
The resource that had been used for implementing Galois field multiplication in environment with ISE software for Xilinx based on behavioral algorithms in Matlab were 28 slices LUTS, 28 LUT-FF and 24 IOB bonded with process time (delay) of 2.236 ns. Figure 6(A) showed the simulation results with ISIM 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

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



(A)



Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	0	82,000	0%
Number of Slice LUTs	28	41,000	1%
Number used as logic	28	41,000	1%
Number using O6 output only	26		
Number using O5 output only	0		
Number using O5 and O6	2		
Number used as ROM	0		
Number used as Memory	0	13,400	0%
Number used exclusively as route-thrus	0		
Number of occupied Slices	8	10,250	1%
Number of LUT Flip Flop pairs used	28		
Number with an unused Flip Flop	28	28	100%
Number with an unused LUT	0	28	0%
Number of fully used LUT-FF pairs	0	28	0%
Number of slice register sites lost to control set restrictions	0	82,000	0%
Number of bonded IOBs	24	300	8%

(B)

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:

$$MSE = \frac{1}{10}((215-215)^2+(183-183)^2+\ldots+(242-242)^2)$$
$$=0$$

The proposed GF2^8 multiplication has been implemented by involving the resources of 28 LUTs with a processing time of 2.236 ns. This GF2^8 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.

No	Parameter	Method in [5]	Method in [6]	Method in [7]2	Method in [9	Proposed Method
1	No Of LUTs	544	43	92	118	28/214
2	Processing Time (ns)	1.6	5.9	3.68	7.16	2.236/2.7

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

[1] Djamalilleil, A. Muslim, M. Salim, Y. Alwi and H. Herman, "Modified Transposition Cipher Algorithm for Images Encryption," The 2nd East Indonesia Conference on Computer and Information Technology (Eiconcit), Makassar, South Sulawesi, (pp. 1-4). Doi: 10.1109/eiconcit.2018.8878326, 2018.

[2] Mu. Annalakshmi & A. Padmapriya, "Zigzag Ciphers: A Novel Transposition Method," IJCA Proceedings on International Conference on Computing and information Technology 2013 IC2IT(2), pp. 8-12, Dec 2013.

[3] P. Poonia & P. Kantha, "Comparative Study of Various Substitution And Transposition Encryption Techniques," International Journal of Computer Applications (0975 – 8887), 145(10), 24-27. Doi:10.5120/ijca2016910783, 2016.

[4] Elaine Barker, "Guideline for Using Cryptographic Standards in the Federal Government:Cryptographic Mechanisms," NIST Special Publication 800-175B Revision 1, DOI: doi.org/10.6028/NIST.SP.800-175Br1, March 2020.

[5] Marwa Sleem, Yousra Alakabani, Ali Rashed and Attif Ibraheem, "Low Power Implementation of AES Mix Columns/ Inverse Mix Column on FPGA," Trans Tech Publications, Switzerland, doi:10.4028/www.scientific.net/AMR.677.311, 2013.

[6] Solmazs Ghaznavi, Catherine Geboty and Reoven Elbaz, "Efficient technique for the FPGA implementation of the AES Mix columns Transformation," International Conference on Reconfigurable Computing and FPGAs, pp.219-224.1, 2009.

[7] Viktor Fischer, Milos Drutarovsky & Pawel Chodowiec, "InvMixcolumn Decomposition and Multilevel Resource Sharing in AES Implementation," in IEEE Trans. On VLSI Systems, 13(8), pp.989-992, 2005.

[8] M. Senthil Kumar and DR. S. Rajalakshmi, "High Efficient Modified MixColumns in Advanced Encryption Standard using Vedic Multiplier,". International Conference on Current Trends in Engineering and Technology, ICCTET, 2014.

[9] Neethan Elizabeth Abraham and Tibin Thomas, "FPGA Implementation of Mix and Inverse Mix Column for AES Algorithm," IJSRD - International Journal for Scientific Research & Development1(9), ISSN (online): 2321-0613, 2013.

[10] C. Nalini, P.V. Anandmohan, D.V. Poornaiah and V.D. Kulkami, "Compact Designs of SubBytes and MixColumn for AES," IEEE International Advance Computing Conference, DOI : 10.1186/s13639-016-0033-y, 2009.

[11] D. Chicco, M.J. Warrens, G. Jurman, "The coefficient of determination R-squared is

more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," PeerJ Computer Science 7:e623, https://doi.org/10.7717/peerj-cs.623, 2021.

[12] Adam Berent, "Advanced Encryption Standard by Example," https://www.adamberent.com/wp-content/uploads/2019/02/AESbyExample.pdf , May 2021.

[13] Ragiel Hadi Prayitno, "Implementation of Modified Encryption and Decrytion Algorithm in Advanced Encrytion Standard into FPGA Device", Disertation, Doctoral Program of Information Technology, Gunadarma University, Jan 2022.