An Optimized S-Box for Advanced Encryption Standard (AES) Design

Oyshee Brotee Sahoo, Dipak K Kole, Hafizur Rahaman

Department of Information Technology
Bengal Engineering and Science University, Shibpur, India
Email: oyshee.sahoo@gmail.com,dipak.kole@gmail.com,hafizur@vlsi.becs.ac.in

***Abstract*—This work presents an optimized Substitution Box (S-Box) for Advanced Encryption Standard (AES) design. The S-Box is one of the most important components of AES. During SubByte transformation, the eight bit input is substituted by eight bit output using the S-Box. S-Box is constructed by composing two transformation – multiplicative inverse in Galois Field $GF(2^8)$ followed by an affine transformation. The S-Box is the most time and power consuming component of AES Algorithm. In this paper, we have modified the affine transformation and tried to reduce the time complexity of AES. The time consumption for AES S-Box is decreased by the proposed affine transformation.**

Keywords-AES, Substitution Box(S-Box), Affine Transformation, Inverse Affine Transformation.

I. INTRODUCTION

Advanced Encryption Standard (AES) specifies the Rijndael Algorithm [2], a symmetric-key block cipher published by National Institute of Standards and Technology (NIST) in 2001[1]. Data Encryption Standard (DES) has been substituted by AES as AES gives higher level of security. AES encrypts and decrypts a data block of 128 bits using the cipher key. The key length can be 128 or 192 or 256 bits. The Substitution Box(S-Box) is the heart of AES Algorithm. It is based on finite field arithmetic.

The AES algorithm is an iterative algorithm composed of different number of rounds for different key size, which is 10 rounds for 128 bits, 12 rounds for 192 bits and 14 rounds for 256 bits. This algorithm consists of four transformations named as SubByte, ShiftRows, MixColumns and AddRoundKey.

The arithmetic of multiplicative inverse and affine transform operation during S-Box creation are based on the arithmetic operations in the finite field $GF(2^8)$[7]. The calculation of this function and its inverse can be done efficiently with combinational logic. This approach has advantages over a straight-forward implementation using read-only memories for table lookups.

The hardware implementation of AES S-Box mainly depends on the efficient hardware implementation of multiplicative inversion in $GF(2^8)$. A hardware implementation of S-BOX has been proposed in [7]. The calculation of this function and its inverse has been done with combinational logic. The resulting circuit ensure low die-size, low transistor count and is efficient for pipelining. This technique is easily adopted within a semi-custom design methodology like a standard-cell design. IBM implementation [10] is also based on a conversion of elements of $GF(2^8)$ into two-term polynomials. ASIC circuits [11] are exploiting more efficient finite field arithmetic.

Most of the hardware implementations of S-Box which are using look-up tables (LUTs) to store the values, need larger area in design. Number of applications which are implied in conventional LUT method, used to generate the multiplicative inverse values as mentioned in [9][10]. The mostly used algorithm for inversion is the extended Euclidean algorithm which is described in [4]. Unfortunately, this algorithm cannot be implemented in hardware. A software implementation of producing multiplicative inverse value was proposed in [8], which is necessary step of S-Box creation and then possibility to implement the process in hardware. This implementation is proposed to embed on small size FPGA with limited memory requirement. This proposed algorithm itself deals with values of finite field $GF(2^8)$ and this can be easily and efficiently implemented in hardware as well as software as it fits perfectly in binary representation and the arithmetic operations are much easier than general number operations [9]. In paper [12], instead of using LUTs for implementing the S-box, logic gate is implemented based on low-complexity composite field for a low-cost S-box for the AES. The improved formulations for the inversion in the sub-fields within the S-box reduce the area complexity of the implementations.

S-Box component consumes much of the total AES circuit power to create it. When the AES circuits are used in low power embedded systems, the reduction of power consumption is a very critical problem. The power consumption of S-Box depends on the number of dynamic hazards. These types of hazards may be prevented by placing the hazard-transparent XOR gates after other gates. Low power S-Box circuit architecture uses a multistage PPRM architecture over composite fields [6]. Taking advantages of dynamic partially reconfigurable of FPGA, an optimal hardware implementation of AES algorithm is presented in paper [13]. In this paper, we propose an improved affine transformation which reduces the time complexity of AES S-Box operation which ultimately helps in speeding up the AES algorithm.

In the remainder of this article, the mathematical preliminaries are briefly described in section II. AES Algorithm is described briefly in section III. The proposed affine transform and inverse affine transform are given in section IV. Section V presents the experimental result. Finally, conclusion is given in section VI.

978-0-7695-4723-7/12 $26.00 © 2012 IEEE
DOI 10.1109/ICACC.2012.35

II. MATHEMATICAL PRELIMINARIES ON GF(2^8)

All the bytes of AES Algorithm are interpreted as a finite field element. Addition and multiplication of finite field elements are quite different from normal rules of addition and multiplication used for numbers.

A. Addition

The Addition of two elements in a finite field is obtained by adding the coefficients of the corresponding powers in the polynomials. Here addition is the simple XOR operation. Addition can be represented as

$$A(x) \oplus B(x) = \sum_{i=0}^{7} a_i x^i \ \oplus \sum_{i=0}^{7} b_i x^i = \sum_{i=0}^{7} (a_i \oplus b_i) x^i$$

Where A(x) and B(x) are the elements of GF(2^8).

Subtraction is identical with addition:

$a(x) - b(x) = a(x) + b(x)$.

B. Multiplication

In the polynomial representation of the elements of finite field GF(2^8), multiplication of two elements corresponds with the multiplication of respective polynomials modulo an irreducible polynomial of degree 8. A polynomial is irreducible if its only divisors are one and itself. For AES algorithm, the irreducible polynomial is

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

or {01}{1b} in hexadecimal notation.

There's a better way for multiplication by using function xtime() – very efficiently multiplies its input by {02}.

Procedure:

For a byte $\{b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0\}$

i. If $b7 = 0$, the result is already in reduced form. Multiplication of the byte by {02} is achieved only by a 1 bit left shift.

ii. If $b7 = 1$, the result is multiplied by x, (that is {00000010} or {02}) can be implemented at the byte level as a 1 bit left shift and a subsequent conditional bitwise XOR with {1b}.

Multiplication by higher powers can be accomplished through repeated application of xtime().

For example, {64} • {23} = {98} because

{64} • {02} =xtime({64}) = {c8}

{64} • {04} =xtime({c8}) = {8b}

{64} • {08} =xtime({8b}) = {0d}

{64} • {10} =xtime({0d}) = {1a}

{64} • {20} =xtime({1a}) = {34}.

thus,

{64} • {23} = {64} • ({01} ⊕{02} ⊕{20})

= {64} ⊕{c8} ⊕{34}= {98}.

III. AES ALGORITHM SPECIFICATION

The AES algorithm is an iterative algorithm consists of four transformations named as SubByte, ShiftRows, MixColumns and AddRoundKey. After an initial Round Key addition, all the four operations are performed serially on the state array for all the rounds except the last round. The MixColumn operation is not performed in the last round. Same is done in reverse during decryption. Fig. 1 and Fig. 2 show the block diagram of encryption and decryption respectively.

The AES Algorithm's operations are performed on a 2-dimentional array of bytes called the State. The State consists of 4 rows, each containing Nb byte, where Nb is the block length divided by 32[1].

The SubByte and InvSubByte transformations are performed through look-up tables (LUTs) operation which is the storage of values for substitution operation. Each element is presented in hexadecimal notation e.g.; {e4}. First hexadecimal number i.e. 'e' is matched with the row index and second hexadecimal number i.e. '4' is matched with the column index of the S-Box table. Corresponding element in the intersection of the row and the column is the desired substituted byte for the element.

The ShiftRows and InvShiftRows transformations shifted the bytes of last three rows of the state cyclically over different number of bytes to left and right respectively. First row is not shifted. Second, third and fourth row is shifted by one byte, two bytes and three bytes respectively.

The MixColumns and InvMixColumns transformation operates on the state column-by-column treating each column as a four term polynomial over GF(2^8) and multiplied modulo $x^4 + 1$ with a fixed polynomial $a(x)$ and $a^{-1}(x)$ respectively, where $a(x)$ and $a^{-1}(x)$ is given by equation 1 and 2 respectively.

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\} \quad (1)$$

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\} \quad (2)$$

In the AddRoundKey transformation, a round key is added to the each column of the state by a simple bitwise XOR operation.

In this paper, we have modified the affine transformation and tried to reduce the time complexity of AES. The time consumption for AES S-Box is decreased by the proposed affine transformation.

IV. PROPOSED AFFINE TRANSFORM AND INVERSE AFFINE TRANSFORM

The SubByte transformation operates independently on each Byte of the State using SBox which is table of 16 rows (0 to F in hexadecimal) and 16 columns (0 to F in hexadecimal), takes input of 8 bits and generates output of 8 bits. An AES SBox is composed of two steps:

1. Calculate the multiplicative inverse over finite field GF(2^8) where the element {00} is mapped to itself.
2. Apply Affine Transformation.

During Decryption only the reverse operation is performed by applying the Inverse Affine Transformation first and then multiplicative inverse.

In the existing AES algorithm the Affine Transformation and Inverse Affine Transformation (over GF(2^8)) is given in equation 3 and 4 respectively.

$$b_i' = b_i \oplus b_{(i+4) mod 8} \oplus b_{(i+5) mod 8} \oplus b_{(i+6) mod 8} \oplus b_{(i+7) mod 8} \oplus c_i \quad (3)$$

$$b_i' = b_{(i+2) mod 8} \oplus b_{(i+5) mod 8} \oplus b_{(i+7) mod 8} \oplus c_i \quad (4)$$

For $0 \le i < 8$, where b_i is the i[th] bit of the byte and $\boldsymbol{c_i}$ is the i[th] bit a constant byte c, which different for both the equation.

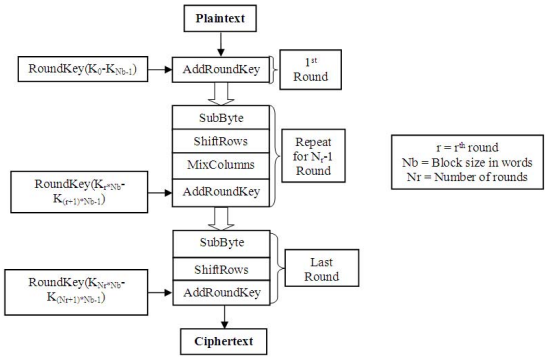

Figure 1. The Encryption process of AES Algorithm

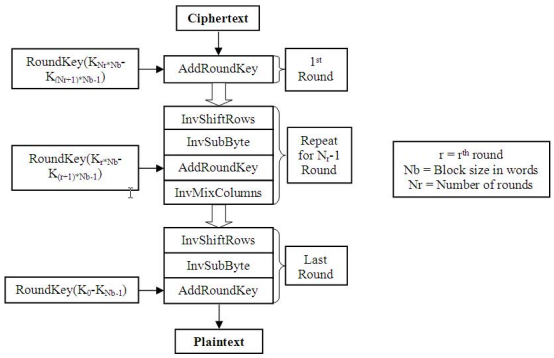

Figure 2. The Decryption process of AES Algorithm

For Affine Transformation the constant c is {63} or {01100011} and for Inverse Affine Transformation the constant c is {05} or {00000101}.

Multiplicative inverse of each element in Galois Field GF(2^8) is unique and an optimized circuit diagram of multiplicative inverse is given in paper of Johannes Wolkerstorfer, Elisabeth Oswald and Mario Lamberger [7].

Our aim is to optimize time for AES algorithm. S-Box is the most time consuming component of AES algorithm. In this paper, we propose Affine Transformation and Inverse Affine Transformation in a new way so that the time taken for S-Box component is less compared to the existing one.

Steps for defining Affine and Inverse Affine transform:

1. We take a linear equation and check whether the equation is a valid affine transform by executing step 2 through step 6.
2. We represent the equation in an 8x8 matrix having 1 in the corresponding element of the equation and all the other elements are 0 in the 1st row. The remaining 7 rows are created by circular shifting the previous row right by 1 bit.
3. First row of the matrix is taken in a form $a_7a_6a_5a_4a_3a_2a_1a_0$.
4. For all possible 8 bit values in between 0 to 255 repeat step 5. Values are represented as $b_7b_6b_5b_4b_3b_2b_1b_0$.
5. If $a_7 \cdot b_0 \oplus a_6 \cdot b_1 \oplus a_5 \cdot b_2 \oplus a_4 \cdot b_3 \oplus a_3 \cdot b_4 \oplus a_2 \cdot b_5 \oplus a_1 \cdot b_6 \oplus a_0 \cdot b_7 = 1$
and $a_7 \cdot b_7 \oplus a_6 \cdot b_0 \oplus a_5 \cdot b_1 \oplus a_4 \cdot b_2 \oplus a_3 \cdot b_3 \oplus a_2 \cdot b_4 \oplus a_1 \cdot b_5 \oplus a_0 \cdot b_6 = 0$
and $a_7 \cdot b_6 \oplus a_6 \cdot b_7 \oplus a_5 \cdot b_0 \oplus a_4 \cdot b_1 \oplus a_3 \cdot b_2 \oplus a_2 \cdot b_3 \oplus a_1 \cdot b_4 \oplus a_0 \cdot b_5 = 0$
and $a_7 \cdot b_5 \oplus a_6 \cdot b_6 \oplus a_5 \cdot b_7 \oplus a_4 \cdot b_0 \oplus a_3 \cdot b_1 \oplus a_2 \cdot b_2 \oplus a_1 \cdot b_3 \oplus a_0 \cdot b_4 = 0$
and $a_7 \cdot b_4 \oplus a_6 \cdot b_5 \oplus a_5 \cdot b_6 \oplus a_4 \cdot b_7 \oplus a_3 \cdot b_0 \oplus a_2 \cdot b_1 \oplus a_1 \cdot b_2 \oplus a_0 \cdot b_3 = 0$
and $a_7 \cdot b_3 \oplus a_6 \cdot b_4 \oplus a_5 \cdot b_5 \oplus a_4 \cdot b_6 \oplus a_3 \cdot b_7 \oplus a_2 \cdot b_0 \oplus a_1 \cdot b_1 \oplus a_0 \cdot b_2 = 0$
and $a_7 \cdot b_2 \oplus a_6 \cdot b_3 \oplus a_5 \cdot b_4 \oplus a_4 \cdot b_5 \oplus a_3 \cdot b_6 \oplus a_2 \cdot b_7 \oplus a_1 \cdot b_0 \oplus a_0 \cdot b_1 = 0$
and $a_7 \cdot b_1 \oplus a_6 \cdot b_2 \oplus a_5 \cdot b_3 \oplus a_4 \cdot b_4 \oplus a_3 \cdot b_5 \oplus a_2 \cdot b_6 \oplus a_1 \cdot b_7 \oplus a_0 \cdot b_0 = 0$
Then the corresponding $a_7a_6a_5a_4a_3a_2a_1a_0$ is the valid Affine transformation.
6. $b_0b_1b_2b_3b_4b_5b_6b_7$ is the first column of the matrix form of the Inverse Affine Transformation. Rests of the columns have come just by shifting cyclically the previous column bottom by 1 bit.
7. First row of the matrix is the corresponding Inverse Affine Transformation.

The new Affine Transformation and Inverse Affine Transformation (over(GF(2^8)) is given by equation 5 and 6 respectively.

$$b_i' = b_i \oplus b_{(i+1)mod8} \oplus b_{(i+7)mod8} \oplus c_i \quad (5)$$

$$b_i' = b_i \oplus b_{(i+2)mod8} \oplus b_{(i+3)mod8} \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus c_i \quad (6)$$

For $0 \le i < 8$, where $\boldsymbol{b_i}$ is the i[th] bit of the byte and $\boldsymbol{c_i}$ is the i[th] bit a constant byte c, which different for both the equation. For Affine Transformation the constant c is {54} or {01010100} and for Inverse Affine Transformation the constant c is {38} or {00111000}.

In matrix form, the affine transform is given by,

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

In matrix form, the Inverse Affine Transformation is given by,

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

The new S-Box formed by applying this new Affine transformation is given in Table I.

V. EXPERIMENTAL RESULT COMPARING WITH EXISTING S-BOX

The proposed S-Box with a different Affine transformation is a time optimized S-Box compared to the existing S-box. Taking the average of some random experiments running on 1.6 GHz. Intel Dual-core machine with 512 MB RAM, the new S-Box is consuming nearly *31 microseconds* where the existing S-Box is consuming nearly *40 microseconds* to create the corresponding S-box with a multiplicative inverse followed by an affine transform.

Then the total AES Algorithm is applied to ten files of different Size files from 1 KB to 10 KB which are created dynamically. The time consumed by SubByte() transformation for each file is observed. It is the time required to fetch the appropriate byte to substitute the comparable byte. The graph in Fig.3 gives the comparative study of the existing and proposed S-Box on the basis of time taken for substitute a byte using the S-Box and inverse S-Box during encryption and decryption respectively for different size of files. In the graph shown in Fig.3, the X-axis represents the size of the files in KB and the Y-axis represents the time taken for byte substitution during encryption and decryption in microseconds.

VI. CONCLUSIONS

This paper has proposed a technique for defining a new affine transform in S-Box operation. A new S-Box with different values is produced. After the software implementation of AES algorithm along with the S-Box, the system time has been calculated experimentally for S-Box, Subbyte transformation during encryption and InvSubbyte transformation during decryption of different files. From the experimental results, it is found that time taken to produce an S-Box is optimized by 10 microseconds approximately. It is shown that the most of the time, the proposed S-Box is taking less time compared to the existing S-Box for different values.

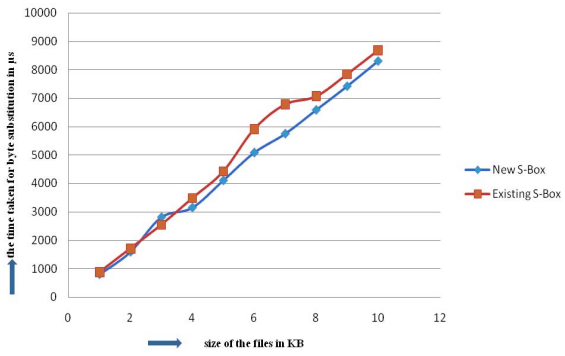

Figure 3. Comparative result of time required to substitute a byte using S-Box for file of different size.

TABLE I. SUBSTITUTION VALUES FOR THE BYTE XY (IN HEXADECIMAL FORMAT)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	54	D7	04	34	ED	8B	64	CE	19	22	BB	75	DD	86	88	FF
1	F2	D3	FE	2C	32	BC	C4	1A	90	8A	AC	67	AB	B4	10	DA
2	07	D1	97	BE	01	25	F9	EA	F6	4F	B1	E1	1C	BA	E2	72
3	36	39	AA	D6	B9	83	CD	B3	3A	91	24	52	76	45	13	F3
4	FD	28	96	4E	B5	9F	B0	5B	6F	CA	7D	E8	82	A9	9A	CB
5	94	9E	D9	6E	A6	2A	1F	4B	70	09	23	3D	0F	17	47	E6
6	65	99	73	C9	2B	EC	15	30	33	3E	2E	DB	98	29	A7	84
7	63	57	27	18	6C	50	C6	0E	D4	FC	4D	5D	66	26	16	92
8	11	E9	6A	95	A4	78	C8	85	35	B8	20	D2	B7	53	42	EB
9	58	DF	1B	55	51	8E	9B	FB	3F	62	3B	89	A2	5F	0A	B6
A	A5	EE	31	F1	03	C1	49	A8	2D	69	6B	AF	60	8F	4A	C3
B	46	05	FA	93	EF	71	E0	7F	68	80	F5	8D	4C	CF	9C	06
C	CC	38	B2	61	56	43	0B	C2	7A	AE	08	5A	F4	2F	F7	0C
D	E7	DE	F0	40	F8	D5	02	1E	A3	0D	7B	C0	3C	21	AD	5C
E	5E	C5	44	9D	7C	41	E3	74	48	A1	C7	81	1D	8C	79	59
F	14	A0	00	12	D8	BD	D0	87	DC	BF	6D	E5	E4	77	37	7E

REFERENCES

[1] NIST, *"Advanced Encryption Standard (AES)"*, FIPS PUBS 197, National Institute of Standards and Technology, November 2001.

[2] J. Daemen and V. Rijmen, *"AES Proposal: Rijndael"*, AES Algorithm Submission, September 3, 1999.

[3] A. Lee, NIST Special Publication 800-21, *"Guideline for Implementing Cryptography in the Federal Government"*, National Institute of Standards and Technology, November 1999.

[4] A. Menezes, P.Van Oorschot and S. Vanstone, *"Handbook of Applied Cryptography"*, CRC press, New York, 1997, p. 81-83.

[5] R. Lidl and H. Niederreiter, *"Introduction to finite fields and their applications"*, Cambridge University Press, Cambridge, 1986.

[6] Sumio Morioka and Akashi Satoh, *"An Optimized S-Box Circuit Architecture for Low Power AES Design"*, IBM Research, Springer-Verlag Berlin Heidelberg, 2003.

[7] Johannes Wolkerstorfer, Elisabeth Oswald and Mario Lamberger, *"An ASIC Implementation of the AES SBoxes"*, Institute for Applied Information Processing and Communication, Graz University of Technology, Springer-Verlag Berlin Heidelberg, 2002.

[8] Naziri, S.; Idris, N.; , *"The memory-less method of generating multiplicative inverse values for S-box in AES algorithm,"*Electronic Design, 2008. ICED 2008. International Conference on , vol., no., pp.1-5, 1-3 Dec. 2008.

[9] R. W. Ward, & T. C. A. Molteno. *"Efficient Hardware Calculation of Inverses in GF(2^8)."* Proceedings of ENZCon'03, University of Waikato, NZ, September 2003.

[10] P. Rudra, K. Dubey, C. S. Jutla, V. Kumar, J. R. Rao, & P. Rahatgi. *"Efficient Implementation of Rijndael Encryption with Composite Field Arithmetic."* Proceedings of the Cryptographic Hardware in Embedded Systems (CHES). pp. 171-184, 2001.

[11] I. Verbauwhede, H. Kuo, *"Architectural Optimization for a 1.82 Gbits/sec VLSI Implementation of the AES Rijndael Algorithm,"* Proceedings of Workshop on Cryptographic Hardware and Embedded Systems, France, Springer-Verlag Berlin Heidelberg 2001.

[12] Mozaffari-Kermani, M.; Reyhani-Masoleh, A.; , "A low-cost S-box for the Advanced Encryption Standard using normal basis," *Electro/Information Technology, 2009. eit '09. IEEE International Conference on* , vol., no., pp.52-55, 7-9 June 2009.

[13] Alaoui-Ismaili, Z.; Moussa, A.; El Mourabit, A.; Amechnoue, K.; , *"Flexible hardware architecture for AES cryptography algorithm,"*Multimedia Computingand Systems, 2009. ICMCS '09. International Conference on , vol., no., pp.438-442, 2-4 April 2009.