Fault Detection and Modification of Substitution Box In Advanced Encryption Standard

R. Prabu, M.Ganesh babu

Abstract: The faults that accidently or maliciously occur in the hardware implementations of the Advanced Encryption Standard (AES) may cause erroneous encrypted/decrypted output. The use of appropriate fault detection schemes for the AES makes it robust to internal defects and fault attacks. In this paper a lightweight concurrent fault detection scheme is presented for the AES. In the proposed approach, the composite field S-box and inverse S-box are divided into blocks and the predicted parities of these blocks are obtained. Through exhaustive searches among all available composite fields, it is found the optimum solutions for the least overhead parity-based fault detection structures. Moreover, through the error injection simulations for one S-box (respectively inverse S-box), and show that the total error coverage of almost 100% for 16 S-boxes (respectively inverse S-boxes) can be achieved. Finally, it is shown that both the application-specific integrated circuit (ASIC) and field-programmable gate-array (FPGA) implementations of the fault detection structures using the obtained optimum composite fields, have better hardware and time complexities compared to their counterparts.

Keywords: AES, composite fields, error coverage, fault detection

I. INTRODUCTION

The paper begins with a brief introduction to the Advanced Encryption Standard, the SubByte and InvSubByte transformation, and finally a short discussion on the previous hardware implementations of the SubByte/InvSubByte transformation.

1.1. The Advanced Encryption Standard

On 2nd January 1997, the National Institute of Standards and Technology (NIST) invited proposals for new algorithms for the new Advanced Encryption Standard (AES). [1] The goal was to replace the older Data Encryption Standard (DES) which was introduced in November 1976 when DES was no longer secure. After going through 2 rounds of evaluation Rijndael was selected and named the Advanced Encryption Standard algorithm on 26th November 2001. [6] The AES algorithm has a fixed block size of 128 bits and a key length of 128, 192 or 256 bits. It generates its key from an input key using the Key Expansion function.

R.Prabu is working as a Assistant professor in Aksheyaa college of Engineering in Department of ECE and M.Ganesh babu is working as Assistant Professor in Valliammai Engineering College in Department of ECE, Email:prabu.6037@gmail.com, ganshbabu87@gmail.com

The AES operates on a 4x4 array of bytes which is called a state. The state undergoes 4 transformations which are namely the AddRoundKey, SubByte, ShiftRow and MixColumn transformation. [4] The AddRoundKey transformation involves a bitwise XOR operation between the state array and the resulting Round Key that is output from the Key Expansion function. SubByte transformation is a highly non-linear byte substitution where each byte in the state array is replaced with another from a lookup table called an S-Box. Shift Row transformation is done by cyclically shifting the rows in the array with different offsets. Finally, Mix Column transformation is a column mixing operation, where the bytes in the new column are a function of the 4 bytes of a column in the state array. Of all the transformation above, the SubByte transformation is the most computationally heavy[3].

1.2. The Subbyte and Invsubbyte Transformation

The SubByte transformation is computed by taking the multiplicative inverse in GF (2^8) followed by an affine transformation. For its reverse, the InvSubByte transformation, the inverse affine transformation is applied first prior to computing the multiplicative inverse. [1] The steps involved for both transformation is shown below.

$$AT(a)=\begin{pmatrix}1&1&1&1&1&0&0&0\\0&1&1&1&1&1&0&0\\0&0&1&1&1&1&1&0\\0&0&0&1&1&1&1&1\\1&0&0&0&1&1&1&1\\1&1&0&0&0&1&1&1\\1&1&1&0&0&0&1&1\\1&1&1&1&0&0&0&1\end{pmatrix}\times\begin{pmatrix}a_7\\a_6\\a_5\\a_4\\a_3\\a_2\\a_1\\a_0\end{pmatrix}\oplus\begin{pmatrix}0\\1\\1\\0\\0\\0\\1\\1\end{pmatrix}\qquad(1.1)$$

SubByte: →Multiplicative Inversion in GF (2^8) → Affine Transformation

InvSubByte: → Inverse Affine Transformation →

$$AT^{-1}(a)=\begin{pmatrix}0&1&0&1&0&0&1&0\\0&0&1&0&1&0&0&1\\1&0&0&1&0&1&0&0\\0&1&0&0&1&0&1&0\\0&0&1&0&0&1&0&1\\1&0&0&1&0&0&1&0\\0&1&0&0&1&0&0&1\\1&0&1&0&0&1&0&0\end{pmatrix}\times\begin{pmatrix}a_7\\a_6\\a_5\\a_4\\a_3\\a_2\\a_1\\a_0\end{pmatrix}\oplus\begin{pmatrix}0\\0\\0\\0\\0\\1\\0\\1\end{pmatrix}\qquad(1.2)$$

Multiplicative Inversion in GF(2^8)

The Affine Transformation and its inverse can be represented in matrix form and it is shown below

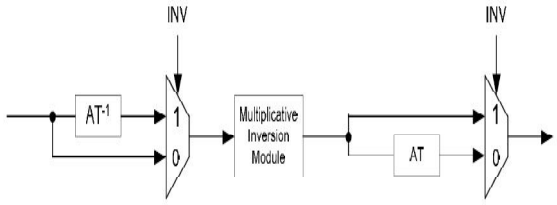


Figure 1.1. Combined SubByte and InvSubByte sharing a common multiplicative inversion module.

The AT and AT-1 are the Affine Transformation and its inverse while the vector *a* is the multiplicative inverse of the input byte from the state array. From here, it is observed that both the SubByte and the InvSubByte transformation involve a multiplicative inversion operation. Thus, both transformations may actually share the same multiplicative inversion module in a combined architecture.

II. S-BOX CONSTRUCTION METHODOLOGY

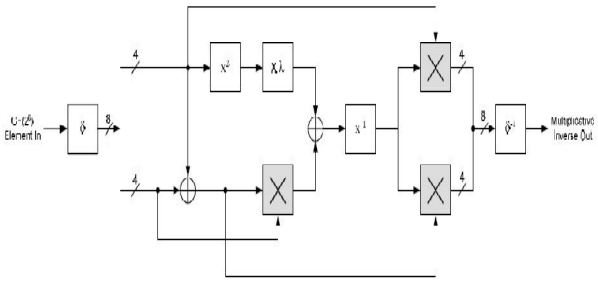
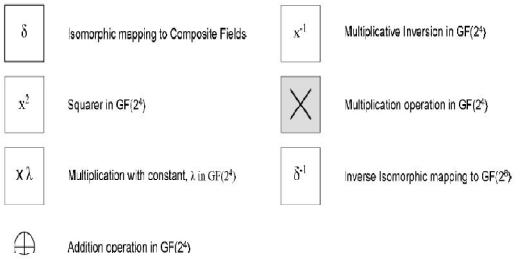


Figure 2.1. Multiplicative inversion module for the S-Box. [1]

The legends for the blocks within the multiplicative inversionmodule from above are illustrated in the below Figure.



2.1. Isomorphic Mapping and Inverse Mapping

The multiplicative inverse computation will be done by decomposing the more complex GF(2^8) to lower order fields of GF(2^1), GF(2^2) and GF((2^2)^2). In order to accomplish the above, the following irreducible polynomials are used.

Where $\varphi = \{10\}_2$ and $\lambda = \{1100\}_2$.

Computation of the multiplicative inverse in composite fields cannot be directly applied to an element which is based on GF (2^8). That element has to be mapped to its composite field representation via an isomorphic function, δ. Likewise, after performing the multiplicative inversion, the result will also have to be mapped back from its composite field representation to its equivalent in GF(2^8) via the inverse isomorphic function, δ-1. Both δ and δ-1 can be represented as an 8x8 matrix. Let *q* be the element in GF(2^8), then the isomorphic mappings and its inverse can be written as δ**q* and δ-1**q*, which is a case of matrix multiplication as shown below, where q_7 is the most significant bit and $q_0$is the least significant bit.

$$GF(2^2) \rightarrow GF(2) \quad : x^2 + x + 1$$
$$GF((2^2)^2) \rightarrow GF(2^2) \quad : x^2 + x + \varphi$$
$$GF(((2^2)^2)^2) \rightarrow GF((2^2)^2) \quad : x^2 + x + \lambda$$

2.2 Composite Field Arithmetic Operations

Arbitrary polynomial can be represented by *bx* + *c* where *b* is upper half term and *c* is the lower half term. Therefore, from here, a binary number in Galois Field *q* can be spilt to q*Hx* + *qL*. For instance, if q = $\{1011\}_2$, it can be represented as $\{10\}_2x$ + $\{11\}_2$, where q*H* is $\{10\}_2$ and q*L*= $\{11\}_2$. q*H* and q *L* can be further decomposed to $\{1\}_2x$ + $\{0\}_2$ and $\{1\}_2x$ + $\{1\}_2$ respectively. The decomposing is done by making use of the irreducible polynomials.

ADDITION IN GF (2^4):

Addition of 2 elements in Galois Field can be translated to simple bitwise XOR operation between the 2 elements.

SQUARING IN GF(2^4):

Let $k = q_2$, where *k* and *q* is an element in GF(2^4),

$$k - \left(\underbrace{k_3k_2}_{k_H}\underbrace{k_1k_0}_{k_L}\right) - k_Hx + k_L - \left(\underbrace{q_3q_2}_{q_H}\underbrace{q_1q_0}_{q_L}\right)^2 - (q_Hx + q_L)^2$$

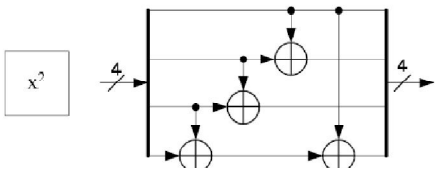
$$k = q_H{}^2x^2 + q_Hq_Lx + q_Hq_Lx + q_L{}^2 = q_H{}^2x^2 + q_L{}^2$$

represented by the binary number of {*k3k2 k1 k0*}$_2$ and {*q3 q2 q1 q0*}$_2$ respectively.

the formula for computing the squaring operation in GF(2^4) is acquired as shown below.

$K_3 = q_3$
$K_2 = q_3 \oplus q_2$
$K_1 = q_2 \oplus q_1$
$K_0 = q_3 \oplus q_1 \oplus q_0$

logic diagram and it is shown below above equation can then be mapped to its hardware



MULTIPLICATION WITH CONSTANT λ:

Let $k = q\lambda$, where $k = \{k3\ k2\ k1\ k0\}_2$, $q = \{q3\ q2\ q1\ q0\}_2$ and $\lambda = \{1100\}_2$ are elements of GF(2^4).

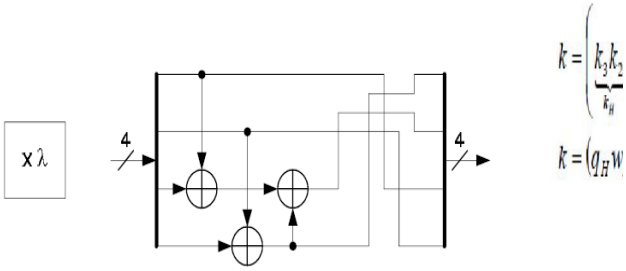
the formula for computing multiplication with constant λ is shown below.

$$K_3 = q_2 \oplus q_0$$
$$K_2 = q_3 \oplus q_2 \oplus q_1 \oplus q_0$$
$$K_1 = q_3$$
$$K_0 = q_2$$

Above equation can then be mapped to its hardware logic diagram and it is shown below

GF(2^4) MULTIPLICATION:

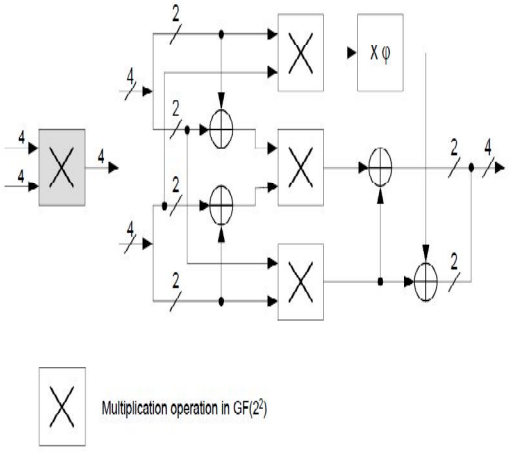
Let $k = q\ w$, where $k = \{k3\ k2k1\ k0\}_2$, $q = \{q3\ q2\ q1\ q0\}_2$ and $w = \{w3\ w2\ w1\ w0\}_2$ are elements of GF (2^4).



Substituting the x^2term with $x^2 = x+\varphi$ yields the following

$K = (q_HW_H)\ (x+\varphi) + (q_HW_L + q_LW_H)x + q_LW_L$

$K = K_Hx+K_L = (q_HW_H + q_HW_L + q_LW_H)x + q_HW_H\ \varphi + q_LW_L \in GF(2^2)$



Above Equation is in the form GF (2^2). It can be observed that there exists addition and multiplication operations in GF (2^2). As mentioned addition in GF (2^2) is but bitwise XOR operation. Multiplication in GF(2^2), on the other hand, requires decomposition to GF(2) to be implemented in hardware.

$$k = \left(\underbrace{k_3k_2}_{k_H}\underbrace{k_1k_0}_{k_L}\right) = k_Hx + k_L = \left(\underbrace{q_3q_2}_{q_H}\underbrace{q_1q_0}_{q_L}\right)\left(\underbrace{1100}_{\lambda_H\ \lambda_L}\right)$$

$k = (q_Hx + q_L)(\lambda_Hx + \lambda_L)$ λ_L can be cancelled out since $\lambda_L = \{00\}_2$.

$k = q_H\lambda_Hx^2 + q_L\lambda_Hx$

GF (2^2) MULTIPLICATION:

Let $k = q\ w$, where $k = \{k1\ k0\}_2$, $q = \{q1\ q0\}_2$ and $w = \{w1\ w0\}_2$ are elements of GF (2^2).

$K = (K_1K_0) = K_1x+K_0 = (q_1q_0)(W_1W_0) = (q_1x+q_0)(W_1x+w_0)$

$K = q_1W_1x^2 + q_0W_1x + q_1W_0x + q_0W_0$

The x^2term can be substituted with $x^2 = x + 1$ to yield the new expression below.

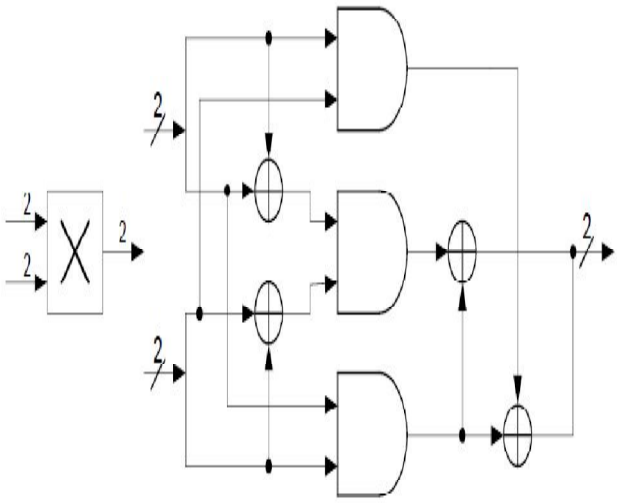
$K = q_1W_1(x+1) + q_0W_1x + q_1W_0x + q_0W_0$

The formula for computing multiplication in GF(2^2) is in below equation

$K_1 = q_1W_1 \oplus q_0W_1 \oplus q_1W_0$

$K_0 = q_1W_1 \oplus q_0W_0$

The equation above can now be implemented in hardware as multiplication in GF (2^2)involves only the use of AND gates.



MULTIPLICATION WITH CONSTANT φ:

Let $k = q\ \varphi$, where $k = \{k1\ k0\}_2$, $q = \{q1\ q0\}_2$ and $\varphi = \{10\}_2$ are elements of GF(2^2).

$K = K_1x + K_0 = (q_1q_0)(10_2) = (q_1x + q_0)(x)$

$K = q_1x^2 + q_0x$

Substitute the x^2 term with $x^2 = x + 1$, yield the expression below.

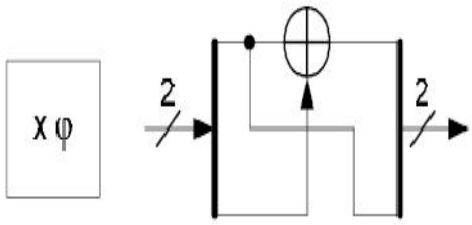
$K = q_1(x+1) + q_0x$

$K = (q1+q0)x + (q1) \in GF(2)$

Above equation is used to derive the formula for computing multiplication with φ and is shown below.

$K_1 = q_1 \oplus q_0$

$K_0 = q_1$



MULTIPLICATIVE INVERSION IN GF (2^4):

A formula has derived to compute the multiplicative inverse of q(where q is an element of GF(2^4)) such that $q^{-1} = \{q_3^{-1}, q_2^{-1}, q_1^{-1}, q_0^{1}\}$.The inverses of the individual bits can be computed from the equation below.

$Q_3^{-1} = q_3 \oplus q_3q_2q_1 \oplus q_3q_0 \oplus q_2$

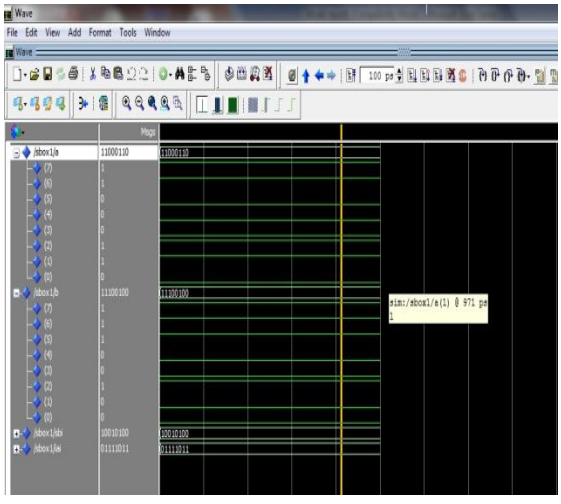
$Q_2^{-1} = q_3q_2q_1 \oplus q_3q_2q_0 \oplus q_3q_0 \oplus q_2 \oplus q_2q_1$

$Q_1^{-1} = q_3 \oplus q_3q_2q_1 \oplus q_3q_1q_0 \oplus q_2 \oplus q_2q_0 \oplus q_1$

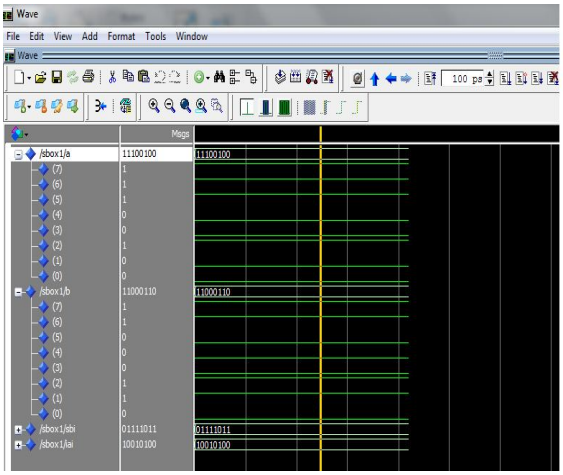
$Q_0^{1} = q_3q_2q_1 \oplus q_3q_2q_0 \oplus q_3q_1 \oplus q_3q_1q_0 \oplus q_3q_0 \oplus q_2 \oplus q_2q_1 \oplus q_2q_1q_0 \oplus q_1 \oplus q_0$

III. ERROR SIMULATIONS

If exactly one bit error appears at the output of the S-box (respectively inverse S-box), the presented fault detection scheme is able to detect it and the error coverage is about 90%. error simulations performed for the S-boxes and the inverse S-boxes using the optimum composite field obtained In our simulations, use stuck-at error model at the outputs of the five blocks forcing one or multiple nodes to be stuck at logic one (for stuck-at one) or zero (for stuck-at zero) independent of the error-free values. and use Fibonacci implementation where, the numbers, the locations and the types of the errors are randomly chosen. In this regard, the maximum sequence length polynomial for the feedback is selected. The injected errors are transient, i.e., they last for one clock cycle. However, the results would be the same if permanent errors are considered. The results of the error simulations using model sim - altera 6.6d (quartus 11.0) and Xilinx ISE version9.1i Simulator (ISim) [2] are presented



S-box output



Inverse S-box output

IV. CONCLUSION AND FUTURE WORK

In this paper a structure independent fault detection scheme is considered for the AES encryption and decryption. This can be applied for both the S-boxes and the inverse S-boxes using lookup tables and those utilizing logic gates based on composite fields. Using S-boxes and inverse S-boxes used for both LUT and composite fields. The proposed scheme has been simulated and its fault coverage has been evaluated in detail. The proposed system is able to find the round and its corresponding transformation in which fault occurred. Multiple Input Signature Register (MISR) detects the presence of error in a device where AES is part of large algorithm. There by optimized hardware is achieved by modifying the structure using sub expression sharing. Hence the reduced number of gates is required in the implementation of AES. The slice overheads are less than those for the other schemes which have the same error coverage. Thus, this scheme has the highest efficiencies, showing reasonable area and time complexity overheads. In future to analyse, design and fault detection in output of the each and every blocks viz 1,2,3and4 ,To detects the faults in the beginning stage ,To avoid the erroneous inputs to the another blocks.

REFERENCES

[1] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "Parity-based fault detection architecture of S-box for advanced encryption standard," in Proc. DFT, pp. 572–580,Oct. 2006.

[2] A.Satoh, T. Sugawara, N. Homma, and T. Aoki, "High-performance concurrent error detection scheme for AES hardware," in Proc. CHES, pp. 100–112, Aug. 2008.

[3] G. Breton, L. Breveglieri, I. Koran, P. Maestri, and V. Piura, "Error analysis and detection procedures for a hardware implementation of the advanced encryption standard," IEEE Trans. Computers, vol. 52, no. 4, pp. 492–505, Apr. 2003.

[4] G. Bertoni, L. Breveglieri, I. Koran, P. Maestri, and V. Piura, "A parity code based fault detection for an implementation of the advanced encryption standard," in Proc. DFT, pp. 51–59, Nov. 2002.

[5] C. H. Yen and B. F.Wu, "Simple error detection methods for hardware implementation of advanced encryption standard," IEEE Trans. Computers, vol. 55, no. 6, pp. 720–731, Jun. 2006.

[6] *R. Karri*, K. Wu, P. Mishra, and K. Yon kook, "Fault-based side-channel cryptanalysis tolerant Irondale symmetric block cipher architecture," in Proc. DFT, pp. 418–426, Oct. 2001.

[7] P. Maistri and R. Leveugle, "Double-data-rate computation as a countermeasure against fault analysis," IEEE Trans. Computers, vol. 57, no. 11, pp. 1528–1539, Nov. 2008.

[8] X. Zhang and K. K. Parhi, "On the optimum constructions of composite field for the AES algorithm," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 53, no. 10, pp. 1153–1157, Oct. 2006.

Prabu R received the B.E. degree in Electronics and Communication Engineering from Thiruvalluvar College of Engineering and Technology which is affiliated to Anna University, Chennai in the year of 2009. And then he received the M.E. degree in VLSI Design from Adhiparaskthi Engineering College, Melmaruvathur in the year 2012. Presently, he is working as a Assistant professor in the department of Electronics and Communication Engineering at Aksheyaa college of Engineering, Puludivakkam. His major research interests focus on wireless communication and VLSI design.

Ganesh Babu M received the B.E. degree in Electronics and Communication Engineering from Valliammai Engineering College which is affiliated to Anna University, Chennai in the year of 2009. And then he received the M.E. degree in Communication Systems from Mepco Schlenk Engineering College, Sivakasi in the year 2011. Presently, he is working as Assistant professor in the department of Electronics and Communication Engineering at Valliammai Engineering College, Kattankulathur. His major research interests focus on image processing, neural networks and OFDM.