ISSN: 2278 – 7798
International Journal of Science, Engineering and Technology Research (IJSETR)
Volume 2, Issue 4, April 2013

High Performance Fault Detection and Correction Scheme for Advanced Encryption Standard

Vinoth Vijay S, Swarna Tharini R, Muthu Ganesh S

***Abstract*—The faults that accidentally 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. The lightweight concurrent fault detection scheme for the AES is used. Through exhaustive searches among all available composite fields, the proposed system uses hamming code and cyclic redundancy check for error detection and correction . Through the error injection simulations for one S-box (respectively inverse S-box), the total error coverage of almost 99% can be achieved. Finally, it is shown that both the application-specific integrated circuit and field-programmable gate-array implementations of the fault detection and correction structures using the obtained optimum composite fields have better hardware and time complexities compared to their counterparts**

***Keywords*: Advanced Encryption Standard, S-Box, inverse S-box, composite field, fault detection.**

I. INTRODUCTION

Symmetric Key Cryptography

Cryptography is a method that has been developed for transfer datas firmly. In digital communications the data is sent through the wire and thus it is not secluded from loss of data. Therefore, secrecy of the transferring data is of severe importance. Encryption is a process which transforms the data that is aimed to be sent to an encrypted data using a key. The encryption process is not confidential but the Secret Key is sent to the receiver. The receiver in turn transforms the received data using the decryption process to obtain the original data. Symmetric key cryptography uses a shared key in encryption and decryption process which can be used for secure long term communications.

The symmetric key cryptography process is shown in Figure 1.1. In this figure, Bob is the sender of the message to Alice over an insecure channel. Eve is the channel eavesdropper who wants to understand the contents of any message sent through the channel. Bob and Alice share the key and Bob's message (plain text) is encrypted using this key. The encrypted message (cipher text) is sent to Alice through the insecure channel.

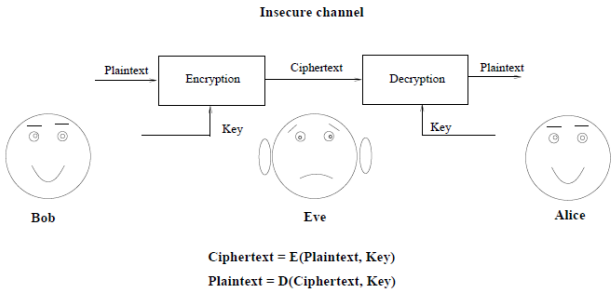


Figure 1.1: Symmetric Key Cryptography.

.

Using the shared key, Alice is able to decrypt the original plain text from encrypted text. It is noted that the need for sharing the key at all times between dispatcher and recipient is a constraint for the symmetric key cryptography, while, it has the advantage of being fast and low computationally intensive in both encryption and decryption.

Symmetric key cryptography comprises two different methods for encryption and decryption. In the first method which is denoted as stream cipher, the bits of data are encrypted/decrypted one at a time. Whereas, in the second method which is called block cipher, blocks of the input data which consist of a number of bits are encrypted/decrypted. The Data Encryption Standard (DES), triple DES (3DES), and the Advanced Encryption Standard (AES) are examples for the block cipher symmetric key cryptography.

II DEVELOPMENT OF THE AES

In 1997, the National Institute of Standards and Technology (NIST) initiated a process to select a symmetric-key algorithm. In 1998, NIST announced the acceptance of fifteen candidate algorithms and requested the assistance of the cryptographic research community in analyzing the candidates. This analysis included an initial examination of the safety and efficiency characteristics for each algorithm. NIST reviewed the results of this preliminary research and selected five final candidates: MARS, RC6, Rijndael, Serpent and Twofish. One can find the comparison of these algorithms. Finally Rijndael algorithm was established among these finalists as the Advanced Encryption Standard. It is noted that before the acceptance of

Rijndael algorithm, DES and its improved variant 3DES were used as symmetric key standards. DES has 16 rounds and encrypts and decrypts data in 64-bit blocks, using a 64-bit key. This can be compared to AES-128 which has 10 rounds where datas are encrypted and decrypted in 128-bit blocks, using a 128-bit key. It is easy to find a comparison between Rijndael and DES and triple DESin [1].

The five finalists in the AES contest mentioned above are iterated block Ciphers. It means that they specify a series of transformations (round) that is Iterated a number of times on the data block to be encrypted or decrypted. Also, Each finalist specifies a method for generating a series of keys from the original user key. This method is called the key schedule, and the generated keys are called round keys., the very first and last cryptographic operations are some form of mixing of round key with the data block are made for each finalist. Such mixing prevents an challenger who does not know the keys from even beginning to encrypt the plaintext or decrypt the cipher text Rijndael has very low RAM and ROM requirements and is very well suited to Restricted-space environments where either encryption or decryption is implemented. It is noted that Rijndael appears to be every time a very good performer in both hardware and software across a wide range of computing environments.

A. Encryption And Decryption Of AES

The AES accepts a 128-bit plain text input. The key can be specified to be 128 (AES-128), 192 or 256 bits. In AES-128, the cipher text is generated after 10 rounds, where, each round consists of four transformations except for the final round which has three transformations. The reverse procedure is used in decryption algorithm transforming the cipher text to the original plain text. Figure 2.1 shows the transformations and rounds in the encryption and decryption of AES-128.

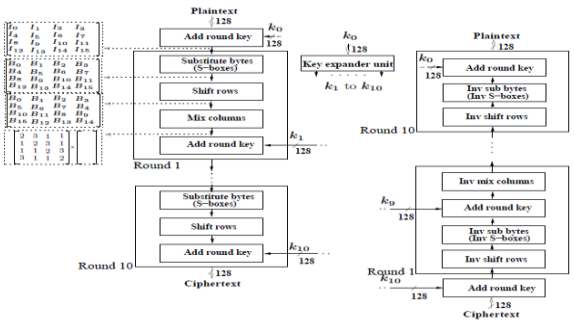


Fig.2.1 Encryption and decryption of AES

B. AES Rounds and Transformations

The four transformations of each round of the encryption each transformation in every round of encryption/decryption acts on its 128-bit input which is considered as a four by four matrix, whose entries are eight bits called state. The transformations in each round of encryption except for the last round are as follows:

• Sub Bytes: The initial transformation in each Step is the bytes substitution, called Sub Bytes, which is implemented by 16 S-boxes. These S-boxes are non-Linear transformations which substitute the 128-bit input state with a 128-bitoutput state. In the S-box, each byte of the state (Ii in Figure 2.1) is substituted by a new byte (Bi in Figure 2.1). S-box will be explained clearly in the next section.

• Shift Rows: Shift Rows is the following transformation in which the four bytes of the rows of the input state are regularly shifted to the left and the first row remains unchanged as shown in the leftmost part of Figure 2.1. The number of left shifts for each row is equal to the number of rows. Denote rows as row is where, i, $0 \le i \le 3$, is the row number. Then, for row 0 no shift, for row1 one shift, for row2 two shifts and for row3 three shifts are required.

• Mix Columns: The third transformation is Mix columns in which each entry in the outputs state are constructed by the multiplication of a column in the input state with a fixed polynomial over GF (2^8). The output states are obtained by multiplying the columns of the input state modulo x4 + 1 with the fixed polynomial of a(x) = (03) x3+ (01) x2+ (01) x+ (02), where the coefficients are in hexadecimal form. The Mix columns matrix representation is shown in Figure 2.1. As seen in figure, the output states are constructed by multiplying the entries of the input states by a fixed matrix whose entries are in the hexadecimal form.

• AddRoundKey: The ultimate transformation is AddRoundKey which XORs the input state with the key of that round, i.e., ki, $0 \le i \le 10$. The AES key expansion unit in Figure 2.1 takes the 128-bit original key, k0, as input and produces a linear array of expanded keys, k1 to k10. Each key is added to the input by 128 two-input XOR gates.

Among the four transformations in the encryption and decryption of AES, only S-box for encryption and inverse S-box for decryption are nonlinear and composite operations. Furthermore, not only is the S-box one of the four round transformations, but it is also used in the key expander unit which generates the keys used in the AES rounds. Therefore, the implementations of these two transformations affect the implementation of the whole AES tremendously. Soon after in this process, the implementation variations of the S-box and inverse S-box including the composite field implementation.

III. S-BOX OPERATION

In cryptography, an S-Box (Substitution-box) is a basic component of symmetric key algorithms which performs substitution operations. In block ciphers, they are typically used to unclear the relationship between the key and the cipher text.

In common, an S-Box takes some number of input bits, m, and transforms them into some number of output bits, n, where n may not be equal to m. An m×n S-Box can be implemented as a lookup table with 2^m words of n bits each. Fixed tables are normally used, as in the Advanced Encryption Standard (AES), but in some ciphers the tables are generated animatedly from the key (e.g.Twofish encryption algorithms).

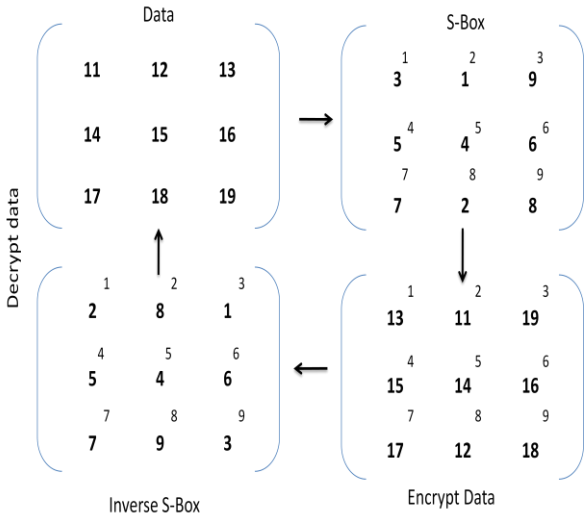


Figure 3.1 Matrix operation of S-box

A. *Encryption of AES in s-box*

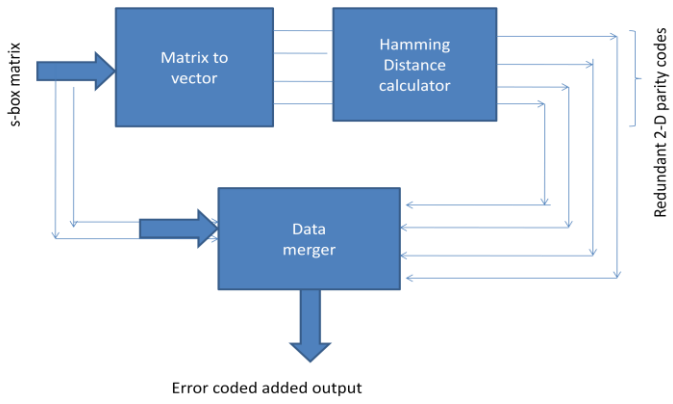


Fig.3.2 Transmitter Section

B. *Decryption of AES in s-box*

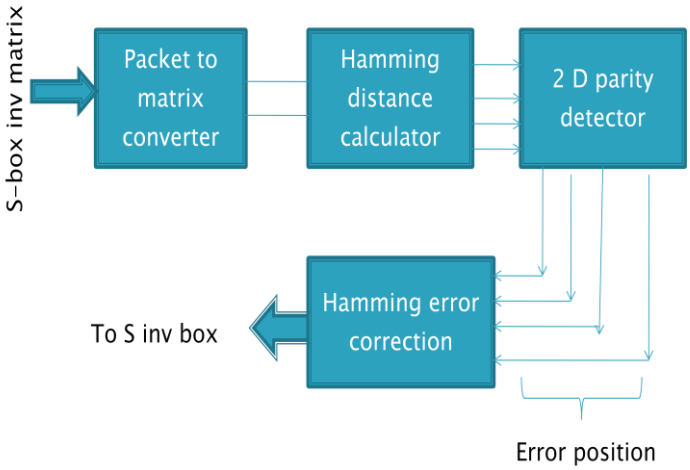


Fig.3.3 Receiver section

IV ERROR CALCULATIONS

If exactly one bit error appears at the output state of the S-box and also in inverse S-box) the presented fault detection scheme is able to detect it and the error coverage is about 100%. This is because in this case, the error indication flag of the corresponding block alarms the fault. However, due to the technological constraints, single stuck-at fault may not be applicable for an mugger to gain more information. Thus, multiple bits will actually be flipped and hence multiple stuck-at errors are also considered in this paper covering both natural faults and fault attacks.[8] For the calculation of the fault coverage for the multiple errors, Pi define as the possibility of error detection in block in Figs. 3.1 and 3. 2. Then, the probability of not detecting the errors in block is (1-pi). For arbitrarily distributed errors in the S-box (respectively inverse S-box), this probability for each block is Independent of those of other blocks. Therefore, one can derive the equation for the error coverage of the randomly distributed errors as

$$EC\% = 100 \times \left(1 - \prod_{i \in S} (1 - p_i)\right)\%$$

Where S is the set of the block numbers where the faults are injected. For randomly distributed errors, the error coverage for each block is Pi ≈ ½. The results of the error simulations are presented in Table I. As shown in the table, using five parity bits

of the five blocks, the error coverage for random faults reaches 97%.Error.

V .HAMMING CODE

When data is transmitted from one location to another there is always the possibility that an error may occur. There are a number of reliable codes that can be used to encode data so that the error can be detected and corrected. With this project you will explore a simple error detection-correction technique called a Hamming Code. A Hamming Code can be use to detect and correct one-bit change in an encoded code word. This approach can be useful as a change in a single bit is more probable than a change in two bits or more bits.

A. *Calculating the Hamming Code*

The key to the Hamming Code is the use of extra parity bits to allow the identification of a single error. Create the code word as follows:

1.Mark all bit positions that are powers of two as parity bits. (positions 1, 2, 4, 8, 16, 32, 64, etc.)

2.All other bit positions are for the data to be encoded. (positions 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.)

3.Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.

Position 1: check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, etc. (1,3,5,7,9,11,13,15,...)

Position 2: check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, etc. (2,3,6,7,10,11,14,15,...)

Position 4: check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, etc. (4,5,6,7,12,13,14,15,20,21,22,23,...)

Position 8: check 8 bits, skip 8 bits, check 8 bits, skip 8 bits, etc. (8-15,24-31,40-47,...)

Position 16: check 16 bits, skip 16 bits, check 16 bits, skip 16 bits, etc. (16-31,48-63,80-95,...)

Position 32: check 32 bits, skip 32 bits, check 32 bits, skip 32 bits, etc. (32-63,96-127,160-191,...etc.

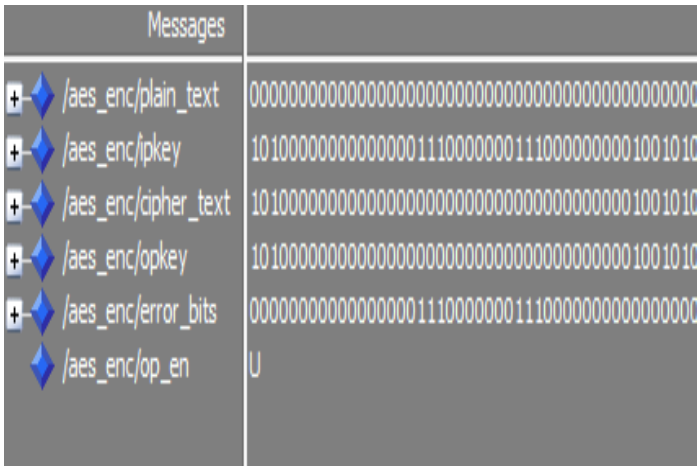
4.Set a parity bit to 1 if the total number of ones in the positions it checks is odd. Set a parity bit to 0 if the total number of ones in the positions it checks is even.

Example:

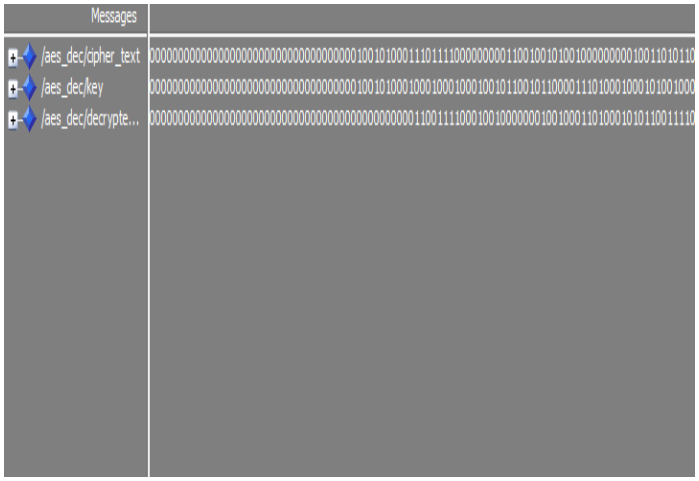
A byte of data: 10011010Create the data word, leaving spaces for the parity bits: _ _ 1 _ 0 0 1 _ 1 0 1 0Calculate the parity for each parity bit (represents the bit position being set):

- Position 1 checks bits 1,3,5,7,9,11: ? _ 1 _ 0 0 1 _ 1 0 1 0. Even parity so set position 1 to a 0: 0 _ 1 _ 0 0 1 _ 1 0 1 0
- Position 2 checks bits 2,3,6,7,10,11:0 ? 1 _ 0 0 1 _ 1 0 1 0. Odd parity so set position 2 to a 1: 0 1 1 _ 0 0 1 _ 1 0 1 0
- Position 4 checks bits 4,5,6,7,12:0 1 1 ? 0 0 1 _ 1 0 1 0. Odd parity so set position 4 to a 1: 0 1 1 1 0 0 1 _ 1 0 1 0
- Position 8 checks bits 8,9,10,11,12:0 1 1 1 0 0 1 ? 1 0 1 0. Even parity so set position 8 to a 0: 0 1 1 1 0 0 1 0 1 0 1 0
- Code word: 011100101010.

B. *Error detection*



C. *Error Correction*



VI CONCLUSION

In this paper, the high performance fault detection and correction scheme for the AES using the S-box and the inverse S-box in composite fields is used. Generally, a number of fault detection scheme are used fault correction scheme is not used in S-box and inverse S-box.in this paper fault detection and correction scheme like Hamming code and CRC is implemented. The implementations show that the optimum S-boxes and the inverse S-boxes using normal basis are more compact than the ones using polynomial basis. The error simulation results show that very high error coverage for the presented scheme. The results of the ASIC and FPGA mapping show that the costs of the presented scheme are reasonable with acceptable post place and route delays.

REFERENCES

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

[2] M. Mozaffari-Kermani and A. Reyhani-Masoleh, "A lightweight concurrent fault detection scheme for the AES S-boxes using normal basis," in Proc. *CHES, Aug. 2008, pp. 113–129.*

[3] S.-Y. Wu and H.-T. Yen, "On the S-box architectures with concurrent error detection for the advanced encryption standard," IEICE Trans. Fundam. Electron.,Commun. Comput. Sci., *vol. E89-A, no. 10, pp.2583–2588, Oct. 2006.*

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

[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] L. Breveglieri, I. Koren, and P. Maistri, "Incorporating error detection and online reconfiguration into a regular architecture for the advanced encryption standard," in Proc. *DFT, Oct. 2005, pp. 72–80.*

[7] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "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.*

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

[9] R. Karri, K. Wu, P. Mishra, and Y. Kim, "Concurrent error detection schemes for fault-based side-channel cryptanalysis of symmetric block ciphers," IEEE Trans. Comput.-Aided Des. *Integr. Circuits Syst., vol. 21, no. 12, pp. 1509–1517, Dec. 2002.*

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

[11] Satoh, S. Morioka, K. Takano, and S. Munetoh, "A compact Rijndael hardware architecture with S-box optimization," in Proc. *ASIACRYPT, Dec. 2001, pp. 239–254.*

S.Vinoth Vijay received his B.E degree in Electronics and Communication Engineering from Jayaraj Annapackiam C.S.I College of Engineering and Technology, Tuticorin Tamil Nadu in 2011 and currently doing M.E-VLSI Design in Madha Engineering College Chennai.

R.Swarna Tharini received her B.E degree in Electronics and Communication Engineering from Dhaanish Ahmed College Of Engineering chennai in 2010 and M.E degree in Applied Electronics from Misrimal Navajee Munoth Jain Engineering College, Chennai in 2012. She is currently working as an Assistant Professor in the department of Electronics and Communication Engineering at Madha Engineering College Chennai.

S.Muthu Ganesh received his B.E degree in Computer Science and Engineering from Sri Subramanya College of Engineering and Technology palani, Tamil Nadu in 2011 and currently doing M.E-VLSI Design in Madha Engineering College Chennai.