ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 4, Issue 5, May 2017

Efficient multiple faults correction in Advanced Encryption Standard (AES)

[1] Anushmita Pathak,[2] Asst. Professor Manjunath C Lakhannavar
[1][2] Department of Electronics & Communication
Ramaiah Institute of Technology,
Bangalore, India.

Abstract - **The data transmitted over the network are vulnerable to electronic eavesdropping. The message transmitted over channels are not similar to the original message since it can be intercepted by some intruder. Encryption is the method to convert the data into secret code and it is viewed as the best approach to guarantee information security. AES is one such encrypting method with the main goal is to achieve secure communication. But its drawback is vulnerability to errors. The cipher text get corrupted once the errors are introduced into the encrypted data, thus resulting in incorrect decrypted data. This paper proposes a method to detect and correct single and correct multiple errors in the data encrypted using AES encryption process.**

Keywords: **AES, Encryption, Eavesdropping,channels.**

I. INTRODUCTION

Encryption is the process of converting the plain text(information) into digital form so that only authorized members can access to it. It is a best approach to guarantee information security. A key, generated by an algorithm is used for encryption and the same key is used for decryption (conversion of encrypted text to original form).

The Advanced Encryption Standard(AES) was established in 2001 by NIST (National Institute of standards and Technology). The block size supported by AES is always 128 bits whereas the key sizes are of 128,192 and 256-bits. AES does not support the extra block sizes.

This paper begins with the overview of the Conventional AES architecture, Weakness of AES, Single detection and correction using Hamming code and Multiple Error correction using Hamming code.

II. CONVENTIONAL AES ARCHITECTURE

AES encryption ,also known as Rijndael Cipher,is symmetric -key algorithm (same key used for both encryption and decryption of data). It supports fixed block size of 128 bits and variable key sizes of 128,192 and 256 bits. The number of rounds are 10,12 and 14 for 128-bit,192-bit and 256-bit keys respectively.

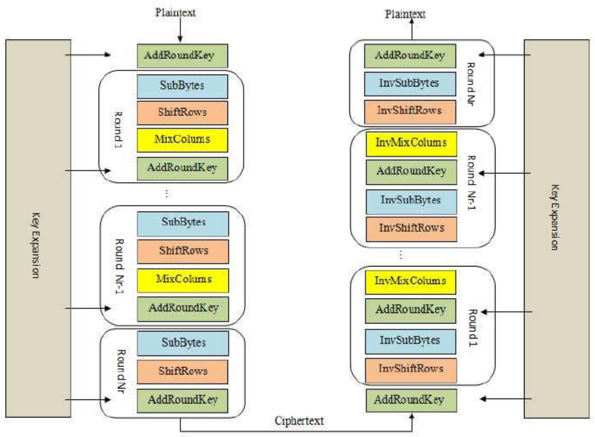


Figure 1: AES Encryption and DEcryptionProcess

In encryption process, as shown in figure 1,each round consists of four stages - Sub byte ,Shift Row,Mix Column and Add Round Key whereas in the Decryption process the inverse of each stages are applied.

In AES, the 128-bit input is arranged in 4 X 4 matrix also known as State. For instance, the 128-bit input is represented as 16 bytes

(d1, d2, d3,….d16),the State is represented as the following:

$$\text{State} = \begin{bmatrix} d1 & d2 & d3 & d4 \\ d5 & d6 & d7 & d8 \\ d9 & d10 & d11 & d12 \\ d13 & d14 & d15 & d16 \end{bmatrix} \qquad (1)$$

The four stages are described as follows:

A. Sub-Byte Transformation: It is a non -linear byte transformation where each byte of the state matrix is substituted by byte of S-box which is already defined. Sbox consists of 256 numbers and corresponding hexadecimal value.It is calculated by first taking the multiplicative inverse of Gallois field GF (28) and undergoing affine transformation.

B. Shift Row Transformation: The bytes in each row of the state matrix are cyclically shifted to the left. The first row is left unchanged whereas the second,third and fourth rows are shifted by one,two and three bytes respectively.

C. Mix Column Transformation: Each column of the state matrix is multiplied by a fixed matrix as shown below:

$$\begin{bmatrix} s1' \\ s2' \\ s3' \\ s4' \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s1 \\ s2 \\ s3 \\ s4 \end{bmatrix} \qquad (2)$$

D. Add Round Key : For each round a subkey is obtained from the user key (128-bits) by using Rijndael's key schedule. Then each byte of the subkey is combined with each byte of the state through bitwise XOR operation.

III. WEAKNESS OF AES

The major issue with AES is that the errors are introduced to the encrypted message during transmission through faulty channels, resulting in incorrect cipher text.

Error correcting code such as Hamming Code is used to detect and correct the single error in the stream of 128-bits data. But it become difficult to detect and correct the multiple errors in the same stream.

Thus, the proposed method is used to correct the multiple errors in data stream of 128 bits.

IV. PROPOSEDMETHOD

A. HAMMING CODE:

The Hamming Code, invented by Richard Hamming in 1950, is linear error-correcting code that can correct single bit error and detect up to two-bit errors. It generalizes Hamming (7,4) code in which 4 data bits are encoded to 7 bits by adding 3 redundancy bits. These extra bits help in detecting and correcting the errors.

The number of redundancy bits to be added to the data bits is given by the following equation:

$$2r \geq m + r + 1 \qquad (3)$$

Where, m = number of data bits

r = number of redundancy bits

The algorithm used in Hamming Code consists of two stages: Encoding and Decoding. In Hamming-Encoding stage, redundancy bits for m-number of data bits are calculated from equation (3) and then they are concatenated with the data bits and are transmitted. Upon receiving the encrypted data on the receiver's side, Hamming-Decoder detects the error, if any, by EXORing the data bits and correcting by using NOT gate. Then the redundancy bits are removed and original data bits are obtained.

B. SINGLE ERROR CORRECTION AND DETECTION

In AES ,the input data stream is of fixed length i.e 128- bits.The cipher text obtained after encryption process is also of the same length.

To detect single error in the encrypted text, 128-bits stream is divided into blocks of 8-bits each and redundancy bits are added to each block.

As per the equation (3), four redundancy bits are obtained for 8-data bits (d1, d2…d8), resulting in 12-bits encrypted text (h1, h2…h12). These redundancy bits are represented as P1, P2, P4 and P8 and are calculated by EXORing the data bits at different locations, as shown below:

p1= d (1) xor d (2) xor d (4) xor d (5) xor d (7)

p2= d (1) xor d (3) xor d (4) xor d (6) xor d (7)

p4= d (2) xor d (3) xor d (4)

p8= d (5) xor d (6) xor d (7)

These redundancy bits are merged with data- bits as shown

below:

1	2	3	4	5	6	7	8	9	10	11	12
p1	p2	d1	p4	d2	d3	d4	p8	d5	d6	d7	d8

The four check bits (y1, y2, y3, y4) are determined from

the above 12-bits data as follows:

y1 = h (1) xor h (3) xor h (5) xor h (7) xor h (9) xor h (11)

y2 = h (2) xor h (3) xor h (6) xor h (7) xor h (10) xor h (11)

y3 = h (4) xor h (5) xor h (6) xor h (7) xor h (12)

y4 = h (8) xor h (9) xor h (10) xor h (11) xor h (12)

The check bits determine whether error is present in the encrypted text and if present, the value of the check bits determine the error location. The error is corrected by the NOT gate.

The above method is applied to all 16-bytes cipher text and it can detect and correct single error only.

C. MULTIPLE ERRORS CORRECTION

For Correcting the multiple errors the same method (as used in single error correction) is used but the position of the error cannot be determined.

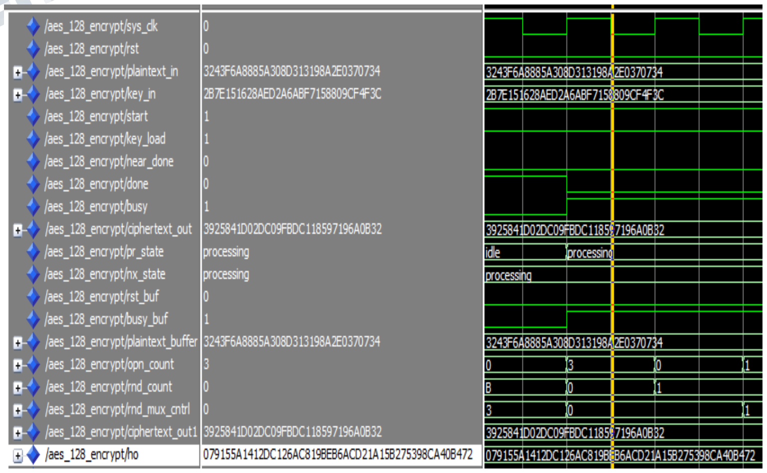
The cipher Text (128-bits) is applied as the input to the hamming encoder, represented as "h". Parity bits are calculated and merged with the data bits of "h" (128-bits), resulting in another stream of 192-bits and represented as "ho".

Multiple Errors are introduced in "ho" and to correct these errors, parity bits are removed and bit-by-bit checking is done between "h" (hamming input) and "ho" (incorrect hamming output) and incorrect bits are corrected by using NOT gate.

V. SIMULATION AND SYNTHESIS RESULT

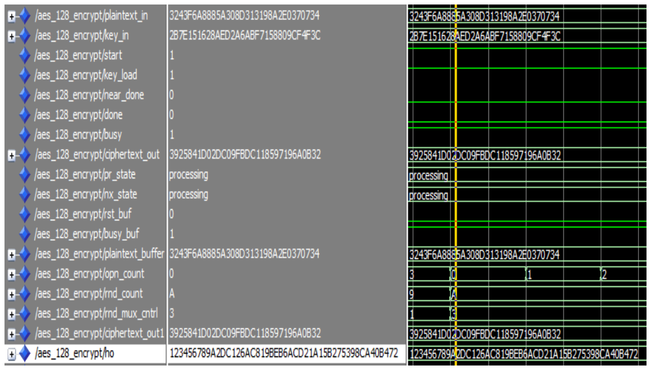
The proposed method was compiled and synthesized on Virtex-7 Xc7v585t FPGA using ISE Navigator and simulated using isim tool.

The results for Single error detection and multiple error correction are given below:



```
/aes_128_encrypt/sys_clk            0
/aes_128_encrypt/rst                0
/aes_128_encrypt/plaintext_in       3243F6A8885A308D313198A2E0370734
/aes_128_encrypt/key_in             2B7E151628AED2A6ABF7158809CF4F3C
/aes_128_encrypt/start              1
/aes_128_encrypt/key_load           1
/aes_128_encrypt/near_done          0
/aes_128_encrypt/done               0
/aes_128_encrypt/busy               1
/aes_128_encrypt/ciphertext_out     3925841D02DC09FBDC118597196A0B32
/aes_128_encrypt/pr_state           processing
/aes_128_encrypt/nx_state           processing
/aes_128_encrypt/rst_buf            0
/aes_128_encrypt/busy_buf           1
/aes_128_encrypt/plaintext_buffer   3243F6A8885A308D313198A2E0370734
/aes_128_encrypt/opn_count          3
/aes_128_encrypt/rnd_count          0
/aes_128_encrypt/rnd_mux_cntrl      0
/aes_128_encrypt/ciphertext_out1    3925841D02DC09FBDC118597196A0B32
/aes_128_encrypt/ho                 079155A1412DC126AC819BEB6ACD21A15B275398CA40B472
```

Figure 2: Single Error Detection And correction



```
/aes_128_encrypt/plaintext_in      3243F6A8885A308D313198A2E0370734
/aes_128_encrypt/key_in            2B7E151628AED2A6ABF7158809CF4F3C
/aes_128_encrypt/start             1
/aes_128_encrypt/key_load          1
/aes_128_encrypt/near_done         0
/aes_128_encrypt/done              0
/aes_128_encrypt/busy              1
/aes_128_encrypt/ciphertext_out    3925841D02DC09FBDC118597196A0B32
/aes_128_encrypt/pr_state          processing
/aes_128_encrypt/nx_state          processing
/aes_128_encrypt/rst_buf           0
/aes_128_encrypt/busy_buf          1
/aes_128_encrypt/plaintext_buffer  3243F6A8885A308D313198A2E0370734
/aes_128_encrypt/opn_count         0
/aes_128_encrypt/rnd_count         A
/aes_128_encrypt/rnd_mux_cntrl     3
/aes_128_encrypt/ciphertext_out1   3925841D02DC09FBDC118597196A0B32
/aes_128_encrypt/ho                123456789A2DC126AC819BEB6ACD21A15B275398CA40B472
```

Figure 3: Multiple Errors Correction

TABLE 1: SYNTHESISED RESULT

Device: Xc7v585t	Proposed Method
No. of Slice Registers	952
No. of Slice LUTS	630
No. of fully used LUT-FF pairs	445
No. of bonded IOBs	391
No. of Block RAM/FIFO	5
Max. Operating Freq.	437 MHz
Power Consumed	529 mW

VI. CONCLUSION

From the result shown in the table-1, the encryption process operates with the maximum operating frequency of 437 MHz.Both the power and area consumed are very less. Also, the proposed method allowed both single and multiple errors correction.

REFERENCES

[1] Abhiram.L. S, Gowrav.L, Punith Kumar.H.L, Sriroop.B.K, Manjunath C Lakkannavar, “FPGA Implementation of Dual Key based AES encryption with Key based S-BOX generation”,

[2] Adham Hadi Saleh, “Design of Hamming Encoder and Decoder Circuits for (64,7) Code and (128,8) Code using VHDL”,Journal of Scientific and Engineering Research,2015.

[3] “Advanced Encryption Standard, Federal Information Processing Standards 197”, National Institute of Standards and Technology, November 2001.

[4] Amandeep Kamboj, R. K Bansal, “High Speed Parallel Concurrent Error Detection Scheme for Robust AES Hardware”, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, October 2013.

[5] Kaijie Wu,Ramesh Karri, “Low Cost Concurrent Error Detection for the Advanced Encryption Standard”, International Journal of Standards and Techniques, May 2010.