# REAL TIME IMPLEMENTATION OF IMAGE CRYPTOGRAPHY USING ADVANCED ENCRYPTION STANDARD

K.Saranya*

Mr.N.Rajesh M.E**

*Abstract*—

**Security in transmission storage of digital images has its importance in today's image communications and confidential video conferencing. Due to the increasing use of images in industrial process, it is essential to protect the confidential image data from unauthorized access. Advanced Encryption Standard (AES) is a well known block cipher that has several advantages in data encryption. However, it is not implemented in real-time applications such as image, video or audio cryptography . In this paper, we present a modification to the Advanced Encryption Standard (MAES) to reflect a high level security and better encryption for image. The modification is done by adjusting the using Block RAM which can be implemented using megawizard function in Quartus IDE. Detailed results in terms of security analysis and implementation are given. Experimental results verify and prove that the proposed modification to image cryptosystem is highly secure from the cryptographic viewpoint.**

**Keywords** – *AES, encryption, decryption, Rijndael, block cipher*

* *M.E VLSI Design, Arunai College Of Engineeering, Tiruvannamalai*

** *Assistant Professor, Arunai College Of Engineering*

## Introduction

In digital world, encryption is emerging as a disintegrable part of all communication networks and information processing systems, for protecting both stored and in transit data. Encryption is the transformation of plain data (known as plaintext) into unintelligible data (known as ciphertext) through an algorithm referred to as cipher. There are numerous encryption algorithms that are now commonly used in computation, but the U.S. government has adopted the Advanced Encryption Standard (AES) to be used by Federal departments and agencies for protecting sensitive information. The National Institute of Standards and Technology (NIST) has published the specifications of this encryption standard in the Federal Information Processing Standards (FIPS) Publication 197.

The AES is a subset of a much larger encryption algorithm known as *Rijndael*, which was one of many proposals to the NIST competing for becoming a standard encryption algorithm. On October of 2000, the NIST announced the *Rijndael* algorithm as the winner due to the best overall score in security, performance, efficiency, implementation capability and simplicity. The AES algorithm is a *symmetric* cipher. In symmetric ciphers, a single secret key is used for both the encryption and decryption, whereas in asymmetric ciphers, there are two sets of keys known as private and public keys. The plaintext is encrypted using the public key and can only be decrypted using the private key.

In addition, the AES algorithm is a *block* cipher as it operates on fixed-length groups of bits (blocks), whereas in *stream* ciphers, the plaintext bits are encrypted one at a time, and the set of transformations applied to successive bits may vary during the encryption process.The AES algorithm operates on blocks of 128 bits, by using cipher keys with lengths of 128, 192 or 256 bits for the encryption process. Although the original Rijndael encryption algorithm was capable of processing different blocks sizes as well as using several other cipher key lengths, but the NIST did not adopt these additional features in the AES

The AES cipher either operates on individual bytes of the State or an entire row/column. At the start of the cipher, the input is copied into the State as described in Section 2.2. Then, an initial Round Key addition is performed on the State. Round keys are derived from the cipher key using the Key Expansion routine. The key expansion routine generates a series of round keys for each round of transformations that are performed on the State.

The transformations performed on the state are similar among all AES versions but the number of transformation rounds depends on the cipher key length. The final round in all AES versions differs slightly from the first Nr transformation performed on the State. Each round of AES cipher (except the last one) consists of all the following transformation:

- SubBytes( )
- ShiftRows( )
- MixColumns( )
- AddRoundKey ( )

The AES cipher is described as a pseudo code in Figure 2. [1] As shown in the pseudo code, all the Nr rounds are identical with the exception of the final round which does not include the MixColumns transformation. The array w[] represents the round keys that are generated by the key expansion routine. In the following sections, individual transformations that are used in each encryption round are described.

Pseudo noise sequences (PN sequences), also referred to as pseudorandom sequences. PN sequence generator block generates a sequence of pseudo random binary numbers using linear feedback shift register (LFSR). Pseudo noise sequence can be used in a scrambler and descrambler. LFSR (linear feedback shift register) is a shift register with a certain number of memory elements. Pseudo noise sequence is a bit stream of ‘0’s and ‘1’s occurring randomly.

Applications of PN sequences include signal synchronization, navigation, radar ranging, random number generation, spread-spectrum communications, multipath resolution, cryptography, and signal identification in multiple-access communication systems.In cryptography, pseudo random noise is a signal which satisfies one or more of the standard tests for statistical randomness

## II. DESCRIPTION OF AES ALGORITHM

The AES algorithm is a symmetric block cipher that can encrypt and decrypt information.

Encryption converts data to an unintelligible form called cipher-text. Decryption of the cipher-text converts the data back into its original form, which is called plain-text.

### *A. AES encryption*

The AES algorithm operates on a 128-bit block of data and executed Nr - 1 loop times. A loop is called a round and the number of iterations of a loop, Nr, can be 10**,** 12, or 14 depending on the key length. The key length is 128, 192 or 256 bits in length respectively.

The first and last rounds differ from other rounds in that there is an additional AddRoundKey transformation at the beginning of the first round and no MixCoulmns transformation is performed in the last round.

In this paper, we use the key length of 128 bits (AES-128) as a model for general explanation.

BLOCK DIAGRAM:

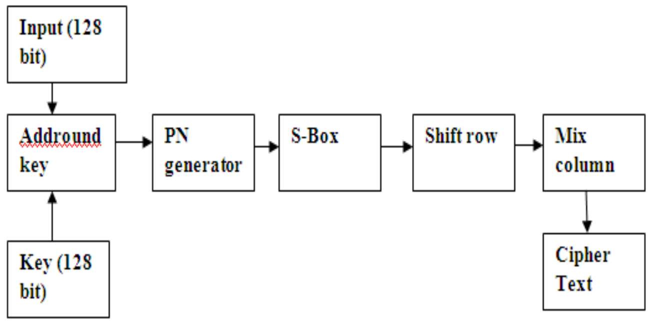

Figure 1 Block diagram for encryption

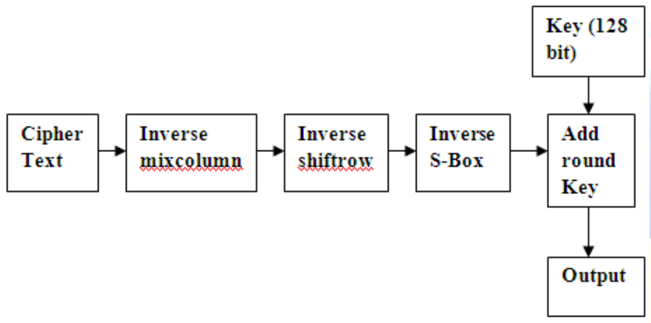

Figure 2 Block diagram for decryption

*SubBytes Transformation:*

The SubBytes transformation is a non-linear byte substitution, operating on each of the state bytes independently. The SubBytes transformation is done using a once-pre- calculated substitution table called S-box. That S-box table contains 256 numbers (from 0 to 255) and their corresponding resulting values. In this design, we use a look-up table as shown in Table 1. This is a more efficient method than directly implementing the multiplicative inverse operation followed by affine transformation.

| | Y | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 1 | 67 | 2b | fe | d7 | ab | 76 |
| | 1 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| | 2 | b7 | fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| | 3 | 4 | c7 | 23 | c3 | 18 | 96 | 5 | 9a | 7 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| | 4 | 9 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| | 5 | 53 | d1 | 0 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| | 6 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 2 | 7f | 50 | 3c | 9f | a8 |
| | 7 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| X | 8 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| | 9 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| | a | e0 | 32 | 3a | 0a | 49 | 6 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| | b | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 8 |
| | c | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| | d | 70 | 3e | b5 | 66 | 48 | 3 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| | e | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| | f | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

Table 1 S-Box

.

*ShiftRows Transformation:*

In ShiftRows transformation, the rows of the state are cyclically left shifted over different offsets. Row 0 is not shifted; row 1 is shifted one byte to the left; row 2 is shifted two bytes to the left and row 3 is shifted three bytes to the left.

*MixColumns Transformation:*

In MixColumns transformation, the columns of the state are considered as polynomials over GF $(2^8)$ and multiplied by modulo $x^4 + 1$ with a fixed polynomial c(x), given by: $c(x)=\{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$.

*AddRoundKey Transformation:*

In the AddRoundKey transformation, a Round Key is added to the State - resulted from the operation of the MixColumns transformation by a simple bitwise XOR operation. The Round Key of each round is derived from the main key using the KeyExpansion algorithm.The encryption/decryption algorithm needs eleven 128-bit RoundKey, which are denoted RoundKey[0] RoundKey[10] (the first RoundKey [0] is the main key).

*B. AES decryption*

Decryption is a reverse of encryption which inverse round transformations to computes out the original plaintext of an encrypted cipher-text in reverse order. The round transformation of decryption uses the functions AddRoundKey, InvMixColumns, InvShiftRows, and InvSubBytes successively.

*AddRoundKey*:

AddRoundKey is its own inverse function because the XOR function is its own inverse. The round keys have to be selected in reverse order. The description of the other transformations will be given as follows.

*InvShiftRows Transformation:*

InvShiftRows exactly functions the same as ShiftRows, only in the opposite direction. The first row is not shifted, while the second, third and fourth rows are shifted right by one, two and three bytes respectively.

*InvSubBytes transformation*:

The InvSubBytes transformation is done using a once-pre- calculated substitution table called InvS-box. That InvS-box table contains 256 numbers (from 0 to 255) and their corresponding values. InvS-box is presented in Table 2.

| | | Y | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
| | 0 | 52 | 9 | 6a | d5 | 30 | 36 | a5 | 38 | bf | 40 | a3 | 9e | 81 | f3 | d7 | fb |
| | 1 | 7c | e3 | 39 | 82 | 9b | 2f | ff | 87 | 34 | 8e | 43 | 44 | c4 | de | e9 | cb |
| | 2 | 54 | 7b | 94 | 32 | a6 | c2 | 23 | 3d | ee | 4c | 95 | 0b | 42 | fa | c3 | 4e |
| | 3 | 8 | 2e | a1 | 66 | 28 | d9 | 24 | b2 | 76 | 5b | a2 | 49 | 6d | 8b | d1 | 25 |
| | 4 | 72 | f8 | f6 | 64 | 86 | 68 | 98 | 16 | d4 | a4 | 5c | cc | 5d | 65 | b6 | 92 |
| | 5 | 6c | 70 | 48 | 50 | fd | ed | b9 | da | 5e | 15 | 46 | 57 | a7 | 8d | 9d | 84 |
| | 6 | 90 | d8 | ab | 0 | 8c | bc | d3 | 0a | f7 | e4 | 58 | 5 | b8 | b3 | 45 | 6 |
| | 7 | d0 | 2c | 1e | 8f | ca | 3f | 0f | 2 | c1 | af | bd | 3 | 1 | 13 | 8a | 6b |
| X | 8 | 3a | 91 | 11 | 41 | 4f | 67 | dc | ea | 97 | f2 | cf | ce | f0 | b4 | e6 | 73 |
| | 9 | 96 | ac | 74 | 22 | e7 | ad | 35 | 85 | e2 | f9 | 37 | e8 | 1c | 75 | df | 6e |
| | a | 47 | f1 | 1a | 71 | 1d | 29 | c5 | 89 | 6f | b7 | 62 | 0e | aa | 18 | be | 1a |

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | b | fc | 56 | 3e | 4b | c6 | d2 | 79 | 20 | 9a | db | c0 | fe | 78 | cd | 5a | f4 |
| | c | 1f | dd | a8 | 33 | 88 | 7 | c7 | 31 | b1 | 12 | 10 | 59 | 27 | 80 | ec | 5f |
| | d | 60 | 51 | 7f | a9 | 19 | b5 | 4a | 0d | 2d | e5 | 7a | 9f | 93 | c9 | 9c | ef |
| | e | a0 | e0 | 3b | 4d | ae | 2a | f5 | b0 | c8 | eb | bb | 3c | 83 | 53 | 99 | 61 |
| | f | 17 | 2b | 4 | 7e | ba | 77 | d6 | 26 | e1 | 69 | 14 | 63 | 55 | 21 | 0c | 7d |

Table 2 Inverse S-Box

*InvMixColumns Transformation:*

In the InvMixColumns transformation, the polynomials of degree less than 4 over GF($2^8$), which coefficients are the elements in the columns of the state, are multiplied modulo ($x^4$+ 1) by a fixed polynomial $d(x) = \{0B\}x^3 + \{0D\}x^2 + \{09\}x + \{0E\}$, where $\{0B\}$, $\{0D\}$; $\{09\}$, $\{0E\}$ denote hexadecimal values.

III Simulation Results

The design has been coded by Verilog HDL. All the results are simulated using on the Quatus II, the Model Sim – 6.6c.

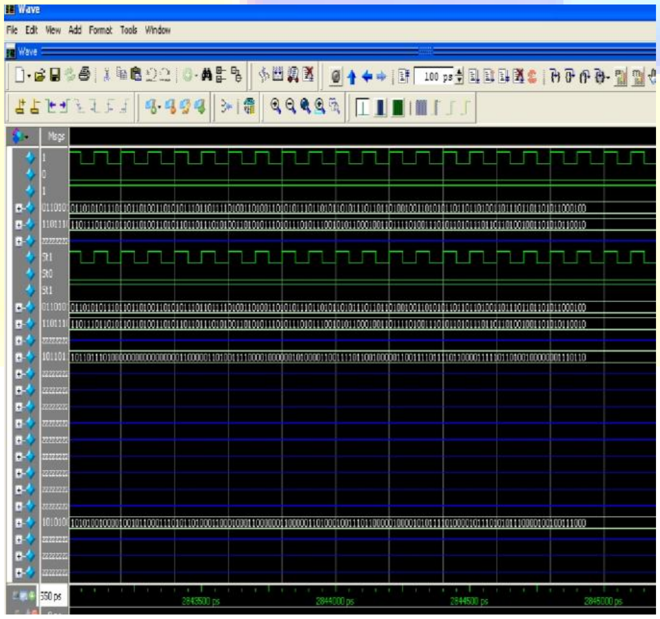

Figure 3 Output for encryption

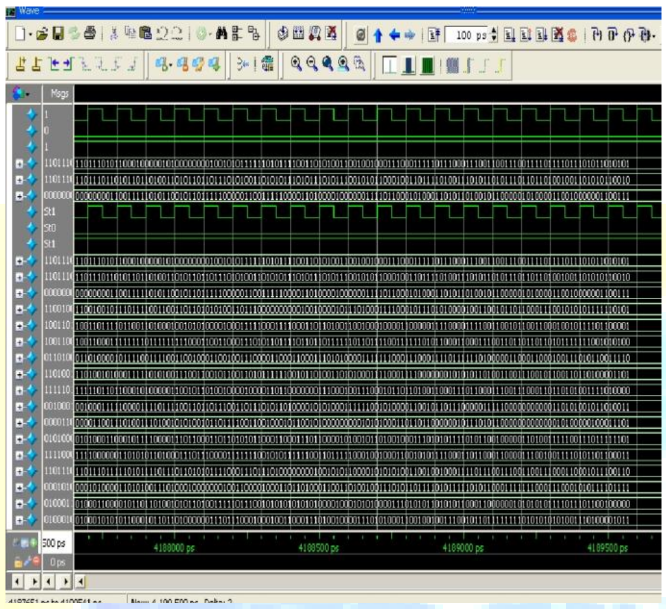

Figure 4 Output for decryption

## IV Conclusion

The AES (Advanced Encryption Standard) architecture for the 128 bit data length and 128-bit key length was designed using Verilog and synthesized with RTL Compiler, physically designed with SOC quartus. With the proposed novel architecture, and fully sub-pipelining, prototyped III FPUA's and then ASIC design was throughput has increased to 58.18 Gbps. The design was prototyped in FPGA's and then ASIC design was made for 90nm Technology. This design has a scope of using in portable devices, where bulk transmission of data is required with high security

## V Future Work

The implementation of combinational s-box circuit to avoid the LUT(Look up table). It will reduce both Static and dynamic power upto 75% and compare to existing system.

REFERENCES

[1]. R. R. Rachh and P. V. Ananda Mohan, "Implementation of AES S-Boxes using combinational logic," in Circuits and Systems,2008. ISCAS 2008. IEEE International Symposium on, 2008, pp. 3294-3297.

[2]. J. Takahashi and T. Fukunaga. Differential fault analysis on AES with 192 and 256-bit keys. IACR eprint archive, 2010-023.

[3].M. Tunstall and D. Mukhopadhyay. Differential fault analysis of the advanced encryption standard using a single fault. IACR eprint archive, 2009-575.

[4].M. Tunstall and D. Mukhopadhyay, "Differential Fault Analysis of the Advanced Encryption Standard using a Single Fault," Cryptology ePrint Archive, Report 2009/575, 2009.

[5] D. Peacham and B. Thomas, " A DFA attack against the AES key schedule," SiVenture White Paper 001, 26 October, 2006.

[5]. Fayed.M,Watheq El-Kharashi.M and Gebali.F,

"A high speed,Fully Pipelined VLSI Architecture for real time AES", 4th International Conference on Information and Communications technology (IClCT),pp 1-2,Dec 2006.

[6] N. Floissac and Y. L'Hyver, "From aes-128 to aes-192 and aes-256, how to adapt differential fault analysis attacks," Cryptology ePrint Archive, Report 2010/396, 2010.

[7]. Z.Xinmiao and K.K.Parhi, "High-speed VLSI architectures for the AES algorithm," Very Large Scale Integration (VLSI) Systems, IEEE Transactions on, vol. 12, pp. 957-967, 2004.