*International Journal of Engineering Research and Development*
***e*-ISSN: 2278-067X, *p*-ISSN: 2278-800X, www.ijerd.com**
*Volume 4, Issue 1 (October 2012), PP. 94-98*

# Image Security Using Modified Advanced Encryption Standard Algorithm (MAES)

Kishore.V[1], Naresh Kumar.B[2], Vasudeva Reddy.T[3]
[1,2,3]*Padmasri Dr.B.V.Raju Institute of Technology, Narsapur, Medak, A.P.India.*

***Abstract—Security in transmission storage of digital images has its importance in today's image communications and confidential video conferencing. Advanced Encryption Standard (AES) is a well known block cipher that has several advantages in data encryption. However, it is not suitable for real-time applications. In this paper, we present a modification to the Advanced Encryption Standard (MAES) to reflect a high level security and better image encryption. The modification is done by adjusting the Shift Row phase. Experimental results verify and prove that the proposed modification to image cryptosystem is highly secure from the cryptographic viewpoint. The results also prove that with a comparison to original AES encryption algorithm the modified algorithm (MAES) gives better encryption results in terms of security against statistical attacks.***

***Keywords—AES, MAES, image encryption, security analysis***

## I. INTRODUCTION

Extensive research has been done in the areas of improving encryption standards, providing better hardware as well as software architecture for implementing security and keeping the cost sensitivity, low disk space and the energy constraints in check. This can be implemented using hardware and software. The advantages of software implementation are portability and flexibility but it offers only limited physical security, especially with respect to key storage and speed. The Data Encryption Standard (DES) which was used earlier for data encryption is being replaced with Advanced Encryption Standard (AES), because DES uses only a very small 56 bit key length. The longer the key length, the more secure is the key. Using larger key length makes more possible keys to search, which makes the algorithm to be more secure. AES has larger key length and has been efficiently implemented both in hardware and software.. This paper primarily focuses on AES encryption algorithm of key length 128 bit, briefs its architecture, and evaluates the performance of the algorithm using software models with slight modifications to improve the security level.

## II. AES ALGORITHM

AES algorithm [2] can be operated in four modes, Electronic Code Block mode (ECB), Cipher Block chain mode (CBC), Cipher feedback (CFB), Output Feedback (OFB). In ECB mode, data block is directly encrypted to form the cipher text and other three modes belong to feedback. ECB mode can be achieved with high speed as it can simultaneously process multiple blocks. Based on this looping structure, the following architecture options were investigated so as to yield optimization of AES algorithm as iterative looping.

The Rijndael AES algorithm[4] uses a symmetric key block cipher both in encryption and decryption. At the start of encryption, input is copied to the State array. State array is nothing but a 4 x 4 matrix of bytes. The encryption algorithm encrypts one block of data at a time to produce the encrypted data block with the use of a secret key. The decryption is simply the reverse of the encryption, and each operation is the inverse of the corresponding one in encryption. The data block length is fixed to 128 bits, while the key length can be 128, 192, or 256 bits. Each data block is rearranged in a matrix form. AES algorithm is an iterative algorithm and each iteration is called a round. The number of bytes in a row of key length is given by equation

$$N = L \big/ 8 \times B_r$$

where, N is the number of bytes, L is the block length in bits, and B**r** is the number of rows in a state array matrix. Each round is iterated 10 times for a 128-bit length key, 12 times for a 192-bit key and 14 times for 256 bit key with 4,6 and 8 bytes in a row of key lengths respectively. Each round uses four transformations and inverses but finalround excludes MixColumn transformations

### A. AES structure[2][7]

AES is an iterated block cipher where it repeats set of transformations called a round transformation :

```
AES (State,CipherKey)
{
KeyExpansion(CipherKey,ExpandedKey);
AddRoundKey(State,ExpandedKey[0]);
For(i=1;i< ;i++)
Round (State,ExpandedKey[i]);
FinalRound (State,ExpandedKey[ ]);
```

}

The round transformation is consisting of four different transformations:
i) The ShiftRow transformation: the rows of the State are cyclically shifted over different offsets.
ii) The MixColumn transformation: the columns of the State are considered as polynomials over GF ($2^8$) and multiplied modulo x4 +1 with fixed polynomial c( x ).
iii) The Round Key addition: a Round Key is applied to the State by a simple bitwise XOR.
iv) The ByteSub transformation: is a non-linear byte substitution, operating on each of the State bytes independently depending on substitution table (or S-box) which is invertible and is constructed by the composition of two transformations:
a) First, taking the multiplicative inverse in GF($2^8$).
b) Then, applying an affine (over GF(2) ) transformation.
A round can be represented as follows:

```
Round(State , ExpandedKey[i])
{
        SubBytes(State);
        ShiftRows(State);
        MixColumns(State);
AddRoundKey(State,ExpandedKey[i]);
}
```

The final round of the cipher is slightly different as below which misses the MixColumns stage:

```
Round(State,ExpandedKey[ ])
{
        SubBytes(State);
        ShiftRows(State);
AddRoundKey(State,ExpandedKey[ ]);
}
```

## III. DATA UNIT AND BLOCKS SPECIFICATION

The data to be encrypted is represented as matrix of bytes called State Matrix. The AES process[3][5] will be initiated & each matrix value is byte valued. After the representation of the input data as a matrix the individual column elements form WORD which is of value 4 bytes. This word representation will be very helpful while going for the key generation which is essential for symetric cryptography.

**A. AddRoundKey**

During each round of an AES process, a separate 128-bit round key is used. The round keys are derived from the cipherkey using a key expansion routine. The AddRoundKey operation is a simple bit-by-bit XOR operation between the state and the round key. An AES process requires a total of Nr+1 AddRoundKey operations, as an additional 'initial' AddRoundKey operation is performed prior to the round operations. For an AES encryption process this initial AddRoundKey operation XORs the plaintext with the cipherkey. There is a need for an additional AddRoundKey operation during an AES process, as there are Nr rounds and Nr+1 subkeys.

**B. SubBytes and InvSubBytes**

The Substitution mechanism[7] in AES is done for each byte & only one table is used for every byte, which means that if two bytes are same, their transformation is also same. The transformation is defined by either a table look-up process or mathematical calculation in the GF($2^8$) field. The substitution box for SubBytes transformation is derived from GF($2^8$) field by taking predefined irreducable polynomial ( polynomial which will have no factors). The contents of the state box are substituted with reference to the SubBytes transformation table as shown in Fig.1.

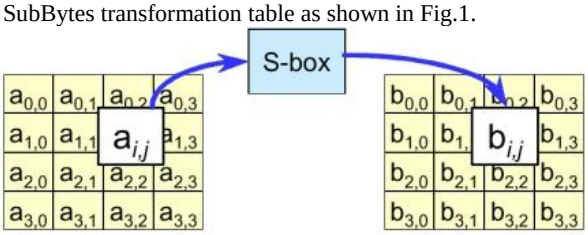


**Fig1.** SubByte Transformation

The irreducible polynomial to derive the elements of the substitution table by GF($2^8$) field defined as $\mathbf{X^8 + X^4 + X^3 + X + 1}$ and as steps the process is discribed for SubByte process.

- Finding the Multiplicative Invers of the Byte value.
- Byte to Matrix conversion.
- Matrix multiplication with constant square matrix X
- Matrix Addition of above result with the column matrix y, which cosists the initial value for combination 00, the 1X1 element in subByte transformation table.

Ex: 63, the 1X1 element

- Matrix to byte conversion.

For the InvSubBytes process steps will be carries in reverse manner with inverse value of the constant matrix X. by modifying 1X1 element or the constant matrix, the respective table byte value will changes. This gives more security in a simple manner for the entire AES process.

**C. ShiftRows and InvShiftRows**

The Shift Rows operation changes the order of bytes in each row of the state. The Row0 is not affected by this transformation. The second row i.e Row1 is shifted cyclically to left by one byte, the Row2 by two and the Row3 by three bytes as seen in Figure2. The inverse operation InvShiftRows is quite similar in style, with only the cyclic shifts made to the right.

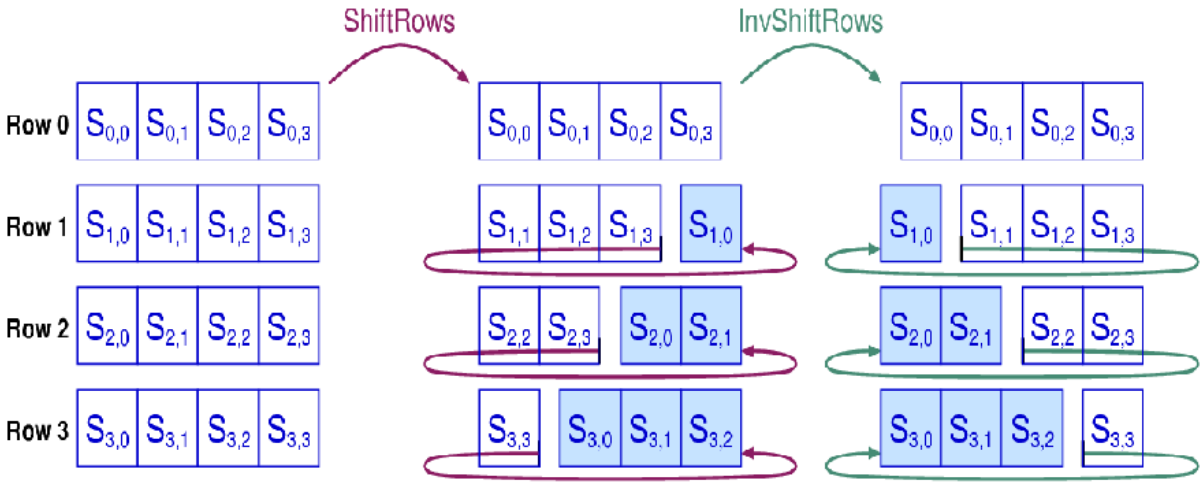


**Fig. 2.** Shift and inverse Shift row transformation

**D. MixColumns and InvMixColumns**

MixColumns is a 32-bit operation that transforms four bytes of each column in the state. The new bytes of the column $S'_{r,c}$ are obtained by the given constant matrix multiplication in GF($2^8$). If the polynomial representation of binary numbers is used, the multiplication process is carried as shown in Fig.3. where Snc (n=0 to 3) represent the original byte column in the state matrix and S'nc(n=0 to 3) represent corresponding modified state matrix column values.

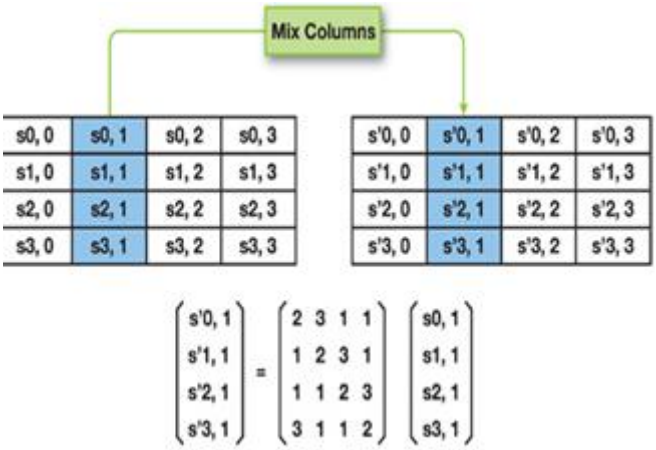


Fig. 3. Mix Columns Representation

## IV. SUBSTITUTION BOX DESIGN[3][7]

The s-box design shows how much the algorithm is cryptographically strong and how the encryptions of data (which can be text or image) have level of security. Changing the initial value i.e. 1st row and 1st column corresponding element, changes entire elements in the state box. In AES, the initial value is taken to be 63.

**A. Transformation using GF ($2^8$) Field**

The byte substitution process either by SubBytes or a mathematical calculation in the GF($2^8$) field is used at the encryption site. In SubBytes to substitute a byte, we interpret the byte as two hexadecimal digits as mentioned earlier. The left digit defines the row and right digit defines column of the substitution table. Although we can use substitution table to find substitution for each byte, AES also defines the transformation algebraically using the GF($2^8$) field with irreducible polynomials x8+x4+x3+x+1.

The SubBytes transformation repeats a routine called *subbyte*, sixteen times, in which multiplicative inverse of the byte is found in GF($2^8$) with irreducible polynomial as the modulus. Note that if the byte is $00_{16}$ its inverse is itself. The inverted byte is then interpreted as a column matrix with the least significant bit at the top and most significant bit at the bottom. this column matrix is multiplied by a constant square matrix, X, and the result, which is a column matrix. This is added with a constant column matrix, y, to give a new byte. note that multiplication and addition are done in GF(2). The *invsubbyte* is doing the same thing in reverse order.

In encryption, multiplication is first and addition is second; In the decryption, subtraction (addition by inverse) is first and division (multiplication by inverse) is second. Let us show how the byte 0C is transformed to FE by subbyte routine and transformed back to 0C by the invsubbyte routine.

***subbyte:***

a. The multiplicative inverse of 0C in GF($2^8$) field is B0 i.e. 10110000
b. Multiplying matrix X by this matrix results in column matrix [c0 c1… c7]
c. The result of XOR operation with initial element 63 (0110 0011) arranging byte in such a way that MSB on the top and LSB at bottom. The result is FE in hexadecimal.

***invsubbyte:***

a. The result of XOR operation will be column matrix
b. The result will be multiplied with matrix $X^{-1}$ and yields to value B0
c. The multiplicative inverse of B0 is 0C

## V. PRINCIPLES OF THE PROPOSED ALGORITHM[4]

In the process of image security using AES cryptographic method, there exist some challenging issues viz. statistical and brute force attacks, memory space requirements and execution speed. To improve the efficiency of the algorithm in the view of security as well as not affecting the speed and the memory space requirements various modifications are carried. Among all these module modifications, the shift row transformation is effective in the sense of fast execution and complexity when compared to the modifications that are carried with other modules..

**A. Modified AES**

**Step 1:** Examine the value of the first Row & first Column, State [0][0] value is even or odd.

**Step 2:** If it is odd, The Shift Rows step operates on the rows of the state matrix; it cyclically shifts the bytes in each row by a certain offset. For MAES, the first & third rows


Fig4.a. Shift Rows Operation if S [0, 0] is ODD

i.e Row0 & Row 2 are unchanged & each byte of Row1 is shifted one to left. Similarly, the Row3 is shifted three to the left respectively as shown in Fig4.a.

**Step 3:** If it is even, the Shift rows step operates on the rows of the stage; it cyclically shifts the bytes in each row by a certain offset. The first & fourth i.e. Row0 & Row3 are unchanged and each byte of Row1 is shifted three to the right. Similarly, the Row2 is shifted by two respectively on to the right as shown in Fig4.b. With this modification in the algorithm required level of security can be achieved.

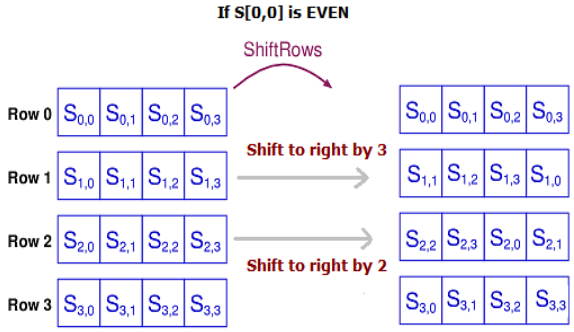


Fig4.b. Shift Rows Operation if S [0, 0] is EVEN

## VI. EXPERIMENTAL RESULTS

a) Original image b) Encrypted c) Decrypted

**Fig.5** Application of the MAES algorithm to plain image/cipher image

## VII. CONCLUSION

In this paper a modified version of AES, namely MAES, is proposed. The modification is done by adjusting ShiftRow phase. The proposed cryptosystem does not require any additional operations rather than the original AES. We have shown that MAES gives better encryption results interms of security against statistical attacks. The secutiy level can be further increased by selecting a simple valid modification in the modules existing. As an example the dynamic S-Box selection for key genertion and Subbyte module operation.

## REFRENCES

[1]. J. Daemen and R.Rijmen, "AES Proposal: Rijndael", version 2, 1999. http://www.esat.kuleuven.ac.be/~rijmen/rijndael.

[2]. "Advanced Encryption Standard(AES)", Federal Information Processing Standards Publication 197, November 26, 2001.

[3]. A. J. Elbirt, W. Yip, B. Chetwynd, and C.Paar, "An FPGA Implementation and Performance Evaluation of the AES Block Cipher Candidate Algorithm Finalist", The Third AES Conference (AES3), NewYork, April 2000,http://csrc.nist.gov/encryption/aes/round2/conf3/aes3papers.html.

[4]. V. Rijmen, "Efficient Implementation of Rijndael S-box", http:// www.esat.kuleuven.ac.be/~rijmen/rijndael.

[5]. J. Daemen and R.Rijmen, "Rijndael: The Advanced Encryption Standard", Dr. Dobb's Journal, pp. 137–139, March 2001.

[6]. B. Schneier, Applied Cryptography: Protocols, Algorithms and Source Code in C. John Wiley and Sons, 1996.

[7]. "Fundamentals of Network Security and Cryptography" by B.Fourozaun, 2Edtn.

[8]. J.J. Amador, R. W.Green "Symmetric-Key Block Cipher for Image and Text Cryptography": International Journal of Imaging Systems and Technology, No. 3, 2005, pp. 178-188.