Modifications to AES Algorithm for Complex Encryption

Priyanka Pimpale, Rohan Rayarikar, Sanket Upadhyay
Students, Department of Computer Engineering,
Thakur College of Engineering and Technology, University of Mumbai, Mumbai, India.

Summary
The Advanced Encryption Standards algorithm was originally proposed by Rijmen and Daemen (Rijndael algorithm) in June 1998. This algorithm is widely accepted due to its strong encryption, complex processing and its resistance to Brute-force attack. The proposed modifications are implemented on the rounds of the algorithm. These modifications enhance the complexity of the encryption process, thereby making it difficult for the attacker to predict a pattern in the algorithm.

Key Words
S Box, Exclusive OR (XOR), Bit, Block, Cipher, Key Matrix, Modified SubBytes, Modified ShiftRows, Modified MixColumns.

1. Introduction

Rijndael Algorithm

The Advanced Encryption Standard comprises three block ciphers, AES-128, AES-192 and AES-256. AES has a fixed block size of 128 bits and a key size of 128, 192, or 256 bits. The block-size has a maximum of 256 bits, but the key-size has no theoretical maximum. The cipher uses number of encryption rounds which converts plain text to cipher text. The output of each round is the input to the next round. The output of the final round is the encrypted plain text known as cipher text. The input given by the user is entered in a matrix known as State Matrix. Following are the four steps.

1.1 SubBytes Step

SubBytes, also known as Byte substitution is the first iterative step of the algorithm in each round. In the SubBytes step, each byte in the matrix is reorganized using an 8-bit substitution box. This substitution box is called the Rijndael S-box. This operation provides the non-linearity in the cipher. The S-box used is derived from the multiplicative inverse over GF (28), known to have good non-linearity properties. To avoid attacks based on simple algebraic properties, the S-box is constructed by combining the inverse function with an invertible affine transformation. The S-box is also chosen to avoid any fixed points (and so is a derangement), and also any opposite fixed points. [5]

1.2 ShiftRows Step

The ShiftRows step is performed on the rows of the state matrix. It cyclically shifts the bytes in each row by a certain offset. The first row remains unchanged. Each byte of the second row is shifted one position to the left. Similarly, the third and fourth rows are shifted by two positions and three positions respectively. The shifting pattern for block of size 128 bits and 192 bits is the same.

1.3 MixColumns Step

In the MixColumns step, the four bytes of each column of the state matrix are combined using an invertible linear transformation [5].
A randomly generated polynomial is arranged in a 4*4 matrix. The same polynomial is used during decryption. Each column of the state matrix is XOR-ed with the corresponding column of the polynomial matrix. The result is updated in the same column. The output matrix is the input to AddRoundKey.

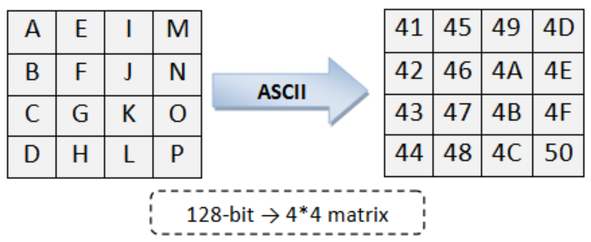
1.4 AddRoundKey

A round key is generated by performing various operations on the cipher key.
This round key is XOR-ed with each byte of the state matrix. For every round a new round key is generated using some operations on the cipher key.

2. Proposed Modified AES Algorithm

The proposed changes in the current AES algorithm are as follows:
The cipher key is arranged in a 4*4 matrix. Each row of this matrix is used for specific operations in the proposed modifications. The first two rows of the cipher key matrix are used in the modified SubBytes round. The third row is used in the modified ShiftRows round. The last row is used in the modified MixColumns round. The AddRoundKey step and Rijndael's key scheduling algorithm in the original AES algorithm remains unchanged. The plain text is thereby arranged in a 4*4 matrix known as State matrix (M).

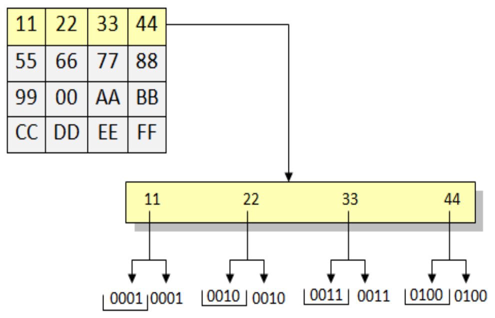
Manuscript received October 5, 2011
Manuscript revised October 20, 2011

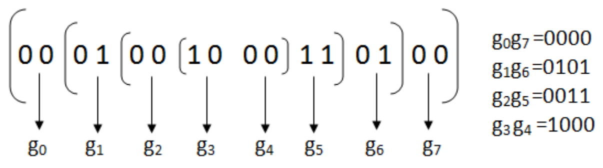


2.1 Modified SubBytes

Converting the first row of the key matrix into its binary equivalent, four groups of 8-bit binary values are generated as shown below. The first 4-bits from each 8-bit value are separated out as shown in figure. These bits are grouped into 4 groups: g0g7, g1g6, g2g5, g3g4 where g0, g1, g2 and g3 represent the row number and g7, g6, g5 and g4 represent the column number of state matrix respectively. The data at location M(g0, g7) is substituted from S-box. The substitution is carried out according to the original SubBytes round of AES algorithm. This is repeated for the remaining locations viz. M(g1, g6), M(g2, g5) and M(g3, g4). Thus, using the first row of the cipher matrix 4 data elements are substituted in the state matrix. Similarly, the entire process is carried out using the second row of cipher key matrix.

- **CIPHER KEY:** 112233445566778899900AABBCCDDEEFF (128-bit cipher key)





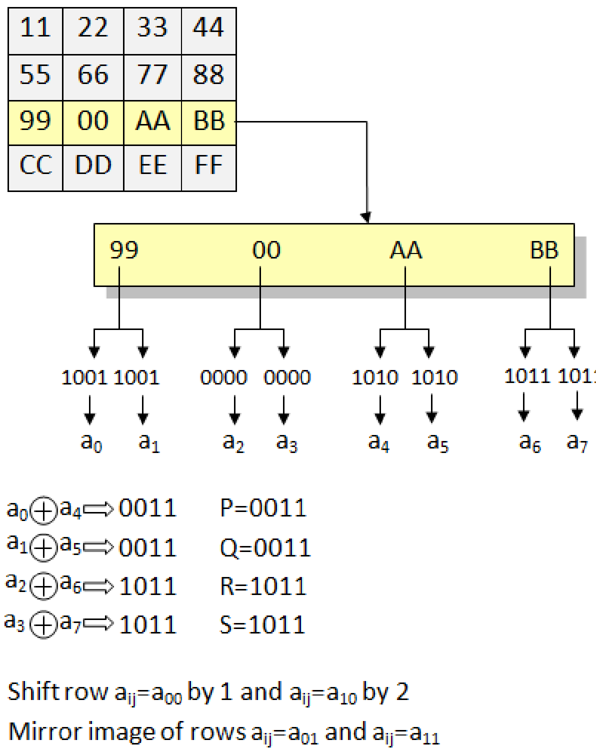
2.2 Modified ShiftRows

In this step the third row of the key matrix is converted to its binary equivalent form. The binary string is grouped into 8 groups, each group having 4 bits starting from first bit as shown in the figure.

The bits of a0 are XOR-ed with those of a4 to obtain a 4 bit binary result, P, as shown above. Similarly, Q, R and S are generated from the remaining groups i.e. [a1,a5], [a2,a6] and [a3,a7] respectively. Bits of P and R are used to identify the row number whereas the bits of Q and S give the number of cyclic left shifts.

The first two bits of P represent the row number which is to be cyclically left shifted. For this row one less than the number of ones in Q is calculated. This gives the number of shifts for the row represented by P. The same process is repeated for R.

Here, if first two bits (say a0 and a1) represent the same row which was previously shifted then a1 and a2 are considered which is the next immediate bit. Again, if the row number is same then a2 and a3 are checked followed by a3 and a1. After checking all the bits, if the row number comes out to be the same, then the same row is shifted by one less than the number of ones in S.



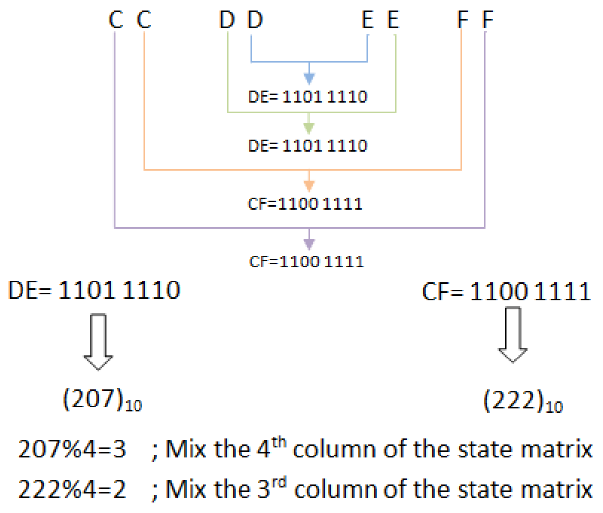
Therefore, the maximum number of rows shifted is 2. Mirror operation is performed on the remaining rows. In this operation, individual data is converted from each row into its binary form and read it from right to left, finally converting it back to hexadecimal value. An example is shown below.

Consider data element 4A (Hexadecimal) = 0100 1010 (binary form)

Reading from right to left we get 0101 0010 = 52 (Hexadecimal)

2.3 Modified MixColumns

In this round the elements of the fourth row of key matrix is grouped as shown in the figure. These groups are then converted into their respective decimal value and modulus-4 operation is performed on each group. The remainder (r) will always lie in range of 0 to 3 i.e. $0 \le r \le 3$. 0, 1, 2 and 3 represent the first, second, third and fourth column respectively. The MixColumn round of the original algorithm is carried out on the selected column. The maximum number of columns mixed in this step is 4.



3. Decryption Process

The decryption process takes place in exactly reverse order of the encryption process. When the encrypted data reaches the receiver, the receiver first XORs the data arranged in the 4*4 matrix with the key matrix. The result obtained is subjected to the following rounds:

3.1 Inverse MixColumns

This process works same as modified MixColumns but in exactly reverse way. The last row of key matrix is considered in this step. The operations are carried out on the data matrix in exactly the reverse order as that of encryption process.

3.2 Inverse ShiftRows

In this round, the bits of the third row of the key matrix are grouped as explained in ShiftRows step mentioned above. It should be noted that the bits are grouped according to the figure shown above. In this round the row pointed by bits in R and the row pointed by bits in P is cyclically shifted to right respectively. The remaining rows are then mirrored randomly in any order.

3.3 Inverse SubByte

In this round, the second row of the key matrix is used. The operations carried out in the SubBytes round as explained above, remains unchanged. The order of substitution is taken from inside to outside which means the coordinates are substituted in order of M(g3, g4), M(g2, g5), M(g1, g6) and M(g0, g7).
We use inverse S-Box for substitution of the data in the matrix.

4. Implementation

The proposed changes to AES algorithm can be implemented in any programming language. Here we have implemented it using java language. The pseudo codes for the step modifications are given below.

4.1 SubBytes Step

```
modifiedSubstituteByte(byte key[][] , byte state[][])
{
    for(number i=0; i<2; i++)
    {
        byte tempkey[8];
        // this loop will get first four bits of selected block //of key matrix
        for(number j=0;j<4;j++)
        {
            // it will store first two bit
            tempkey[j*2]=first & two bit of key[i][j];
            // it will store next two bit
            tempkey[(j*2)+1]=third & fourth bit of key[i][j];
        }
        for(number j=0;j<4;j++)
        {
            substitute (tempkey[j],tempkey[7-j]) byte of state matrix using S-Box ;
        }
    }
}
```

4.2 ShiftRows Step

```
modifiedShiftRows(byte key[][] , byte state[][])
{
    byte tempkey[4];
    number count[2];
    for(number i=0;i,2;i++)
    {
        temp[i*2]=XOR(first four bits of key[2][0],first four bits of key[2][2]);
        temp[(i*2)+1]=XOR(next four bits of key[2][0],next four bits of key[2][2]);
        count[i]=countNumberOfOnes(temp[(i*2)+1]);
}
for(number i=0;i<2;i++)
{
number row=findRow(temp[i*2]);
cyclicLeftShift(row , count[i]);
}
number remainingRows[]=findRemainingRows();
doMirrorRow(remainingRows);
}
```

4.3 MixColumns Step

```
modifiedMixColumn(byte key[][] , byte state[][])
{
byte tempkey[8];
for(number j=0;j<4;j++)
{
tempkey[j*2]=first four bits of key[3][j];
tempkey[(j*2)+1]=remaining four bits of key[3][j];
}
for(number i=0;i<4;i++)
{
byte temp=tempkey[i],tempkey[7-i];
mix the (temp%4) column of state matrix with the
randomly generated polynomial
}
}
```

5. Strength

Even though the modifications are performed on the original AES algorithm, the security of the original algorithm remains intact. The cipher key in AES is of 128 bits. Therefore to break the cipher key it requires 2128 possibilities and tests to be carried out. This is theoretically almost impossible. Therefore, the Brute-force Attack fails on the AES algorithm.
The modifications made to the existing AES have made sure that there is no fixed pattern in any of the steps of the algorithm. The modifications have provided the algorithm with strong diffusion and confusion. Therefore, statistical analysis of the ciphertext fails.
The most important security advantage is that no differential or linear attacks on AES have been able to break the algorithm.

6. Conclusion

The modifications proposed by us can be implemented without increasing the size of the key block. Though the original algorithm is very secure the proposed changes in the processing of the algorithm will help to encrypt the data by making stronger diffusion and confusion. It also increases the complexity of the algorithm multiple times. These proposed changes once implemented will make it very difficult to decrypt the ciphertext without proper decryption key.

7. References

[1] J.Daemen and V.Rijmen, AES Proposal: Rijndael, NIST's AES home page, http ://www:nist:gov/aes.

[2] "Announcing the Advanced Encryption Standard (AES)", Federal Information Processing Standards Publicatrion 197,November 2001

[3] Xinmiao Zhang and Keshab K. Parhi, "Implementation Approaches for the Advanced Encryption Standard Algorithm", 1531-636X/12, IEEE 2002.

[4] Dr (Mrs) Pushpa R. Suri and Sukhvinder Singh Deora, "Design of a modified Rijndael algorithm using 2D Rotations", IJCSNS International Journal of Computer Science and Network Security, VOL.11 No.9, September 2011.

[5] Advanced Encryption Standard, http://en.wikipedia.org/wiki/Advanced_Encryption_Standard.

Priyanka Pimpale

Final Year B.E (CE).Student

Thakur College of Engg. & Tech.,

University of Mumbai, Mumbai, India.

Rohan Rayarikar

Final Year B.E (CE).Student

Thakur College of Engg. & Tech.,

University of Mumbai, Mumbai, India.

Oracle Certified Java SE6 Programmer

Sanket Upadhyay

Final Year B.E (CE).Student

Thakur College of Engg. & Tech.,

University of Mumbai, Mumbai, India.

Oracle Certified Java SE6 Programmer