Design of a Modified Rijndael Algorithm Using 2D Rotations

Pushpa R. Suri†, Sukhvinder Singh Deora††

†Associate Professor, Department of Computer Sc. & Applications, Kurukshetra University, Kurukshetra, India
††Assistant Professor, Dept. of Computer Sc. & Applications, NC Institute of Computer Sciences, Israna, Panipat India

Summary
Rijmen and Daemen had proposed Rijndael algorithm for Advanced Encryption Standard (AES) in June 1998. It was accepted for commercial use due to its symmetric and parallel structure, well adapted to modern processors and its suitability to smart cards. Rijndael algorithm uses an iterated block cipher with a variable block length and a variable key length. The block length and the key length can be independently specified to 128, 192 or 256 bits. We are proposing a modification which we rotate the bytes using 2 Dimensional rotation of the block after Step 4, thereby increasing confusion-diffusion. The proposed scheme will have improved complexity that increases the security for a 128 and 256 bits block case without increase in the length of the key used.

Key words:
Exclusive OR (XOR), permutation, bytes, substitution, shift and 2D rotation operation, S-box

1. Introduction

AES named as Advance Encryption Standard (AES), is based on the Rijmen and Daemen submitted Rijndael algorithm in June 1998. Rijndael was announced to be the final selection for AES in October 2000. AES is an iterative block cipher with a variable length and a variable key length. It is based on some very simple operations like Exclusive OR (XOR), bits shifts and permutations of the columns [1 & 2]. The algorithm in its original form contained four transformations in each rounds, namely, Byte Substitution, Shift Row, Mix Columns, Add (XOR) Round Key. The last round does not use the Mix Columns transformation. Chun Yen et. al. improved Rijndael algorithm and constructed a new S-box. After the improvement, iterative output cycle is 256, and algebraic expression reached 254 items [3]. In this paper we present a modification in Rijndael algorithm, by adding one more round that increases the overall confusion-diffusion of the bytes thereby increasing the complexity which cryptanalysis of the algorithm. The proposed transformations involve mathematical opeartaions that are easy to implement in software level especially by using MATLAB programming due to matrix-based structure of the algorithm.

2. Theoretical Consideration

In the original Rijndael algorithm, the plaintext blocks and keys can be arranged in variable sizes (16, 24 or 32 bytes). The logical view of the block structure is shown in Fig 1. Unlike the DES, the AES algorithm has a highly mathematical structure. There are four steps (see fig. 2) involved in the algorithm, which perform specific transformations in the input plaintext. The algorithm can be used with three key lengths (independent of selected block length): 128, 192, or 256 bits. It can consist of 10, 12 or 14 rounds where each round consists of transformations as discussed below.

	Column			
Row	1	2	3	4
1	A	B	C	D
2	E	F	G	H
3	I	J	K	L
4	M	N	O	P

Fig. 1 Initial plaintext/key block

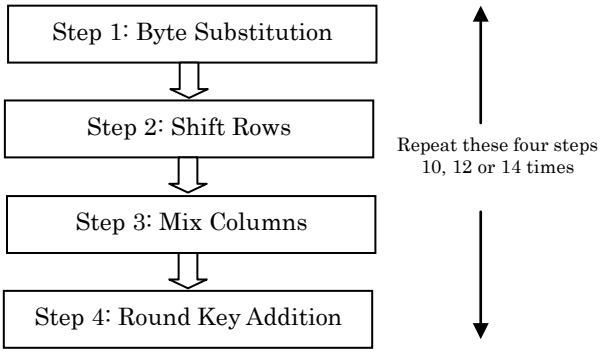

Fig. 2: Steps in Rijndael's algorithm

2.1 Byte Substitution

In this transformation, operation based on the secret S-box is applied to each byte separately. The inverse of the byte substitution are in the inverse table that has a mapping of the inverses at the corresponding locations in the S-box.

Manuscript received September 5, 2011
Manuscript revised September 20, 2011

2.2 Shift Row

The rows of block are cyclically shifted in this transformation. Table 1 shows the number of byte shifts in Row 0 is zero and in the subsequent rows; Row 1, Row 2 and Row 3 is dependent on the number of columns in the block, say Nc.

Table 1 Shift Policy while Encryption

Nc	Byte Shifts		
	Row 1	Row 2	Row 3
4	1	2	3
8	1	3	4

Inverse of this transformation is simple cyclic shift in the reverse direction or applying (Nc-Byte Shits) in the same cyclic shift.

2.3 Mix Column

In this transformation, every column is multiplied with a fixed polynomial say c(x), i.e. c(x) ⊕ m(x) is the resultant value of each byte value. Inverse of this transformation is again a mix column where multiplication is done with the multiplicative inverse of c(x).

2.4 Round Key Addition

A round key derived using some operations on the cipher key is XORed with the entire block state obtained till the Mix Column transformation. The same round key is the inverse of it and can be XORed during the decryption round.

3. Proposed Modified Algorithm

In our modified approach to the Rijndael's algorithm, the plaintext blocks and keys can be arranged in any square variable size (4X4, 4X8 etc). Instead of having four steps, we are proposing five transformations in a round approach. In addition to the regular four transformations, the fifth transformation has been placed at Step 5 (see Fig. 3). Hence the modified AES will contain following transformations; Byte Substitution, Shift Rows, 2D Rotate Block, Mix Column and Round Key Addition. Step 1, Step 2, Step 3 and Step 4 as in the four step original algorithm already discussed, and Step 5 involving our newly introduced "2D Rotate block" transformation.

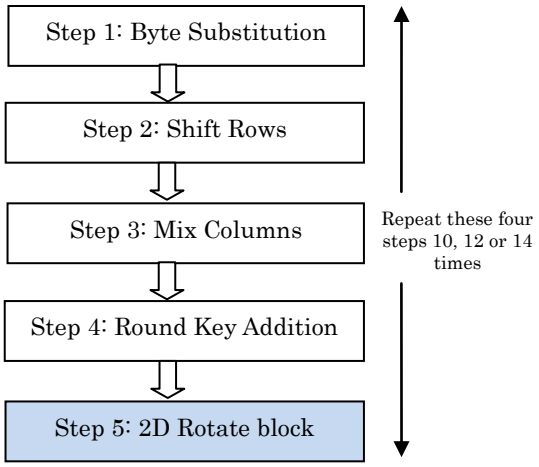

Fig. 3 Modified Rijndael's algorithm

4. 2D Rotate Block Explained

This new kind of rotation we have introduced in the original Rijndael algorithm has been taken from our previous work on 3D Array Block ciphers [4]. In this kind of rotation, the entire 2D Array block is rotated by certain angle depending upon certain value of the key bits. We have suggested the same approach in AES using certain mathematical operations on the matrix structure of the block. Its use will further increase the confusion aspect in the information bytes when transformed to the ciphertext.

4.1 Notations Used

We denote two operations on matrices, rCOOM(M) that denotes reverseColumnsOrderOfMatrix M i.e. function that arranges the columns of the matrix in reverse order and rROOM(M) that denotes reverseRowsOrderOfMatrix M i.e. function that arranges the rows of the matrix in reverse order. Also we use the standard notation M' in our discussions to denote the transpose of a matrix M.

4.2 Encryption Phase

In this step, the entire block is rotated in clockwise direction by an amount of 0°, 90°, 180°, 270° during encryption depending upon the 2 bits of the key value 00,01,10,11 respectively (refer Table 2).

Table 2 Rotation Policy while Encryption

Key bit value	Rotation (Clockwise direction)
00	0°
01	90°
10	180°
11	270°

This will change the relative positioning of the information in the 2D array. The modified step can be implemented by use of the mathematical operations as discussed below:
If we consider the initial array of the bytes as a 2D array M of size say nXn, where n is the number of rows and number of columns of the array then a rotation of 90° can be thought of as a two step mechanism.
i.e. Rotation of 90° = rCOOM (M') (1)
where reverseColumnsOrderOfMatrix() function only arranges the columns of the matrix in reverse order (as shown in Fig. 4). Consider for example that we are having a 4X4 matrix M then the 90° rotation can be considered as follows:

M =

Row \ Column	1	2	3	4
1	A	B	C	D
2	E	F	G	H
3	I	J	K	L
4	M	N	O	P

M'=

Row \ Column	1	2	3	4
1	A	E	I	M
2	B	F	J	N
3	C	G	K	O
4	D	H	L	P

rCOOM (M')=

Row \ Column	4	3	2	1
1	M	I	E	A
2	N	J	F	B
3	O	K	G	C
4	P	L	H	D

Fig. 4 90° Rotation of the 2D Array

Similarly, reversing the order of the columns of the Matrix M (as shown in Fig.5) is equivalent to a 180° rotation. Hence a function rCOOM (M) operation can be used in this case,

i.e. Rotation of 180° = rROOM (rCOOM (M)) (2)

M =

Row \ Column	1	2	3	4
1	A	B	C	D
2	E	F	G	H
3	I	J	K	L
4	M	N	O	P

rCOOM (M)=

Row \ Column	4	3	2	1
1	D	C	B	A
2	H	G	F	E
3	L	K	J	I
4	P	O	N	M

rROOM (rCOOM (M))=

Row \ Column	4	3	2	1
1	P	O	N	L
2	L	K	J	I
3	H	G	F	E
4	D	C	B	A

Fig. 5 180° Rotation of the 2D Array

and lastly a rotation of 270° can be done by reversing the row order of the transposed matrix M (as shown in Fig. 6). Mathematically, it can be denoted by the operations as discussed below:

i.e. Rotation of 270° = rROOM (M') (3)

The operations defined above can be applied using MATLAB very easily or one may use any other programming language as it requires two basic types of operations, transpose and reversal of order of rows / columns only.

M =

Row \ Column	1	2	3	4
1	A	B	C	D
2	E	F	G	H
3	I	J	K	L
4	M	N	O	P

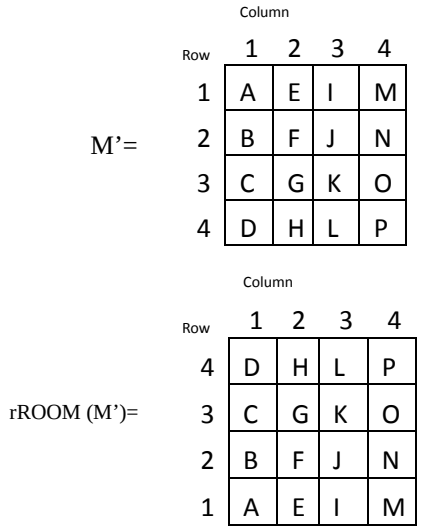

Fig. 6 270° Rotation of the 2D Array

4.3 Decryption Phase

It is noteworthy that the operations during the decrypting phase can be carried out in two different manners. In the first approach, one may use the same bits but rotate the elements in anti-clockwise direction at an angle that was used while encryption (refer Table 3). But for this approach one has to define the decryption operations separately.

Table 3 Rotation Policy while Decryption (anti-clockwise rotation case)

Key bit value	Rotation (anti-Clockwise direction)
00	0°
01	90°
10	180°
11	270°

In the second approach, one may use in the anti-clockwise direction. OR another alternative way is to rotate the matrix in clock-wise direction but for some different angle as specified below:

In this approach one needs to find the 2's complement of the key bit values of rotation and add 1 to it. During addition operation, carry in the 4's place may be ignored. The rotations in clockwise direction using computed bits, is given in the Table 4 for the encryption process. It can be easily verified that the rotations are obtained by adding 1 to the 2's complement of rotation bits, ignoring last carry, for clockwise rotation functions only. Moreover, the operations to be carried out in these cases are same as discussed in above discussions.

Table 4 Rotation Policy while Decryption (clockwise rotation)

Key bit value	2's Complement+1 (ignore carry)	Rotation (Clockwise direction)
00	11+1=00	0°
01	10+1=11	270°
10	01+1=10	180°
11	00+1=01	90°

5. Key Usage

Naim Ajlouni et.al. had suggested a new approach in Key Generation and Expansion in Rijndael Algorithm. Use of random pool of keys has been suggested that simplifies the process of generating and expanding cipher key for the algorithm [5].

However, for this modified form, there is no need of increasing the length of the key. The same old round key (Step 4 of the original AES algorithm) can be used to generate two parity values. The column values of the round key can be added to produce the two bit parity code of the two bytes separately. These two bits can be used to rotate the block as discussed in Section 4. Thus we have introduced another step / transformation without increasing the key length which is advantageous.

In case of a 4X4 block, the 2D Block rotation can be used as explained in the 2D Block Rotation Explained (Section 4). However, in case of a 4X8 block, one can consider it as a special case of two 4X4 blocks. The round key can be used in the same way for the two 4X4 sub-blocks.

5.1 Backward Compatibility

In order to provide backward compatibility in the modified AES implementation, one can take two parity bits after each round as '00'. Since '00' represents a 0° rotation during encryption and decryption phases, the modified algorithm will work as the original AES.

5.2 Key Transfer

As the parity bit sum is used to rotate the block cipher by certain angle, the transfer of the key becomes very important. We confirm the use of our 3D Parity Bit Structure for increasing error free rate of transfer of key bits [6].

6. Implementation

The proposed design is complex to be implemented using hardware only. We suggest the use of software based implementation, preferably in MATLAB due to intrinsic

design of the language in line with the operations/transformations required in the modified AES. However, one may also implement the design discussed above in languages like C also.

```
modifiedAES(byte in[4*Nc], byte out[4*Nc], word w[Nc * (Nr+1)]){
        byte state[4, Nc];
        state = in;
        AddRoundKey(state, w[0, Nc-1])

        for(roundNo=1; roundNo<Nr; roundNo++){
                SubBytes(state);
                ShiftRows(state);
                MixColumns(state);
                AddRoundKey(state, w[Nr*Nc,(Nr+1)*Nc-1]);
                2DRotateState(state, parityCode(state));
        }
        SubBytes(state);
        ShiftRows(state);
        AddRoundKey(state, w[Nr*Nc], (Nr+1)*Nc-1);
                2DRotateState(state, parityCode(state));

        out=state;
}
```

Fig. 7 Pseudo-code for Modified AES

7. Strength

The original AES was having the complexity of the order of the bits used as key. The expected strength is of the order of 2^{127} for 16 bytes of key and 2^{255} for 32 bytes of key. With the introduction of the new round the complexity of the intruder will increase by the order of $3^{\text{NumberOfRounds}}$. This is because at the end of each round in the modified form of AES, the attacker needs to check for 3 possible block values that could have resulted due to 2D Array block rotation. However, this step has made the AES turned to a Feistel structure.

If we denote the time taken in a single 2D Rotation as 't' and '3t' in case of predicting using brute-force attack. Then the time complexity during encryption increases by (NumberOfRounds)Xt. However, in case of decryption, one needs go by (3t) $^{(\text{NumberOfRounds})}$ time complexity.

8. Conclusions

We have provided modification of AES that can be applied without increase in the size of the key block. Inclusion of one more round has further increased the complexity involved to decrypt ciphertext of AES using Brute-force attack. Backward compatibility provided in the modified approach will be beneficial till systems in communication do not upgrade to the software implementation of the same. In all, one can conclude that AES turned to a Feistel structure might prove more complex to be attacked by intruders with malified intensions.

9. 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] Chun Yan,Yanxia Guo, "A Research and Improvement Based on Rijndael Algorithm", 2009 First International Conference on Information Science and Engineering, Nanjing, Jiangsu China, December 26-December 28, ISBN: 978-0-7695-3887-7

[4] Dr. (Mrs) Pushpa R. Suri, "A Cipher based on 3D Array Block Rotation", International Journal of Computer Science and Network Security, VOL.10 No.2, February 2010, pp. 186-191.

[5] Naim Ajlouni et.al., "A new approach in Key Generation and Expansion in Rijndael Algorithm", The International Arab Journal of Information Technology, Vol. 3, No. 1, January 2006, pp. 35-41.

[6] Dr. (Mrs) Pushpa R. Suri, "3D Parity Bit Structure: a novel technique to correct maximal number of bits in a simpler way", International Journal of Computer Science and Internet Security, VOL.9 No.8, August 2011, pp. 182-186.

Pushpa R. Suri is Associate Professor in the Department of Computer Science and Applications at Kurukshetra University, Haryana, India. She has supervised a number of PhD students in the area of Network Security and Cryptograhic techniques. She has also published a number of research papers in National and International Journals and Conference Proceedings. Her areas of interest are Data Structures, Network Security & Cryptography.

Sukhvinder Singh Deora holds the degrees of M.C.A., M.Phil. in Computer Science and is working as Assistant Professor in N.C. Institute of Computer Sciences, Israna, Panipat, India. He is also a Research Scholar at Department of Computer Science and Applications, Kurukshetra University, Kurukshetra, India. This work is a result of his research in the topic of Network Security. To his credit are many prominent papers in the area of data security and issues related to it, published in eminent Journals of India. He has edited Proceedings of National Level Seminars/Conferences. He also has an industry experience of 1.5 years in the areas of Testing, Java and Database design issues. His interest areas include Network Security, Theoretical Computer Sciences, Data Structures, S/W Testing and Database Designing. He is also a permanent member of Indian Society of Information Theory and Applications (ISITA).