Full Description of New Advanced Encryption Standard

By: Morteza Abdolrahim Kashi
University of Manitoba
Canada-Winnipeg

1. Introduction

This document is to describe Rijndael the new Advanced Encryption Standard. I show the mathematics knowledge needed to understand this procedure then we explain the design model of the Rijndael .Then I explain implementation of the cipher and the reverse of the cipher. Then the advantage of this procedure is explained and we discus the limitations existed in this procedure of encryption .We discus the ways this method can be extended .I will introduce the references for this kind of encryption at the end of this paper.

2. Mathematics requirements

We need to know many operations which use bytes to do the operations where bytes are Showing elements in the field GF (2 power 8). We have to consider other operations in terms of 4 bytes word. Here we explain the basic mathematics needed to implement Rijndael.

2. A Polynomial representation

To show a polynomial we consider a byte b is made from bits b0, b1, b2, b3, b4, b5, b6, b7 .We can consider this byte to show a polynomial as follow:

$$b_0 + b_1 x + b_2 x^2 + b_3 x^3 + b_4 x^4 + b_5 x^5 + b_6 x^6 + b_7 x^7$$

For example a byte that has a hexadecimal value of 47 has a binary value of 01000111

This represents the following polynomial:

$b_6x^6 + b_2x^2 + b_1x + b0$

2. A.1 Addition of polynomials

When we want to add two polynomials together we just add corresponding coefficients in each polynomial in mod 2 . For example we have the following equation:
$(x^7 + x^5 + x^4 + x^3 + x^2 + x + 1) + (x^6 + x^5 + x^4 + 1) = x^7 + x^6 + x^3 + x^2 + x$

As we can see the binary coefficients in the first polynomial is 10111111 and the binary coefficient for the second polynomial is 01110001 and the resulting polynomial has the binary coefficient of 11001110. Clearly each bit in the binary coefficient of the result is the result of XOR function of corresponding bits in each polynomial. For example we have (1+1=0, 0+1=1, 1+0=1 and 0+0=0)

2. A.2 Multiplication

The multiplication is done in two phase. The first phase looks like the ordinary algebra multiplication. We multiply each term of the first polynomial by each term of the second one. Then we do addition operation as discussed before .For example we can have the

$A(x) = x^6 + x^4 + x^3 + x^2 + 1 \qquad B(x) = x + 1 \qquad C(x) = A(x)B(x)$
following multiplication:

$(x^6 + x^4 + x^3 + x^2 + 1)(x + 1) = (x^7 + x^6) + (x^5 + x^4) + (x^4 + x^3) + (x^3 + x^2) + (x + 1)$
$= x^7 + x^6 + x^5 + x^2 + x + 1$
If we have:

$(a_3x^3 + a_2x^2 + a_1x + a_0)(b_3x^3 + b_2x^2 + b_1x + b_0) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c0 = C(x)$

Where:

$c_0 = a_0.b_0 \quad c_1 = a_1.b_0 \oplus a_0.b_1 \quad c_2 = a_2.b_0 \oplus a_1.b_1 \oplus a_0.b_2 \quad c_3 = a_3.b_0 \oplus a_2.b_1 \oplus a_1.b_2 \oplus a_0.b_3$

$c_4 = a_3.b_1 \oplus a_2.b_2 \oplus a_1.b_3 \quad c_5 = a_3.b_2 \oplus a_2.b_3 \quad c_6 = a_3.b_3$

Obviously C(x) can not be shown by a 4-byte vector. By changing C(x) modulo a

Polynomial of degree 4, we can reduce the degree to the degree below 4. When we are in Rijndael algorithm we must do that by the polynomial:

$$M(x) = x^4 + 1$$

So we will have the modular product A(x) by B(x) denoted by C(x) is defined as follow:

$$C(x) = A(x) \otimes B(x) = c_3 x^3 + c_2 x^2 + c_1 x + c_0$$

Where:

$$c_0 = a_0.b_0 \oplus a_3.b \ \oplus a_2.b_2 \oplus a_1.b_3 \qquad c_1 = a_1.b_0 \oplus a_0.b_1 \oplus a_3.b_2 \oplus a_2.b_3$$
$$c_2 = a_2.b_0 \oplus a_1.b_1 \oplus a_0.b_2 \oplus a_3.b_3 \qquad c_3 = a_3.b_0 \oplus a_2.b_1 \oplus a_1.b_2 \oplus a_0.b_3$$

If we write this equation in the form of matrix multiplication we will have:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} . \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

Let multiply B(x) by x. Then we have:

$$b_3 x^4 + b_2 x^3 + b_1 x^2 + b_0 x$$

We can calculate $x \otimes B(x)$ by reducing above result by

$$x^4 + 1 \qquad \mathrm{mod}\, ule$$

Then we will have:

$$b_2 x^3 + b_1 x^2 + b_0 x + b_3$$

So we can show the multiplication by *x* using the matrix form as follow:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 00 & 00 & 00 & 01 \\ 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \end{bmatrix} . \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

3. Specification

The Rijndael cipher is a kind of cipher that has iterated block cipher plus a variable block length and a variable key length. Numbers of 128, 192, and 256 bits can be used to assign to the length of the block and the length of the key. We explain the cipher structure in this section.

3.1 The State, the Cipher Key and the number of rounds

We refer to the state as all transformations that operate on the intermediate result and the intermediate cipher result is called a state .We can use a rectangular array of bytes to show a State. We have 4 rows in this array, and we write **Nb** to show the number of columns and can be calculated by dividing the block length by 32.We show all of above explanations in the following figure where the state has **Nb**=6 and the cipher key has **Nk**=4.Sometimes these blocks are shown in the case of one-dimensional arrays consisting 4-byte vectors, when each vector includes the corresponding column in the rectangular array representation .Then the length of these arrays are 4, 6 or 8 respectively and indices in the ranges 0..3, 0..5 or 0..7. 4-byte vectors will sometimes be referred to as words.

A0,0	A0,1	A0,2	A0,3	A0,4	A0,5
A1,0	A1,1	A1,2	A1,3	A1,4	A1,5
A2,0	A2,1	A2,2	A2,3	A2,4	A2,5
A3,0	A3,1	A3,2	A3,3	A3,4	A3,5
A4,0	A4,1	A4,2	A4,3	A4,4	A4,5

K0,0	K0,1	K0,2	K0,3
K1,0	K1,1	K1,2	K1,3
K2,0	K2,1	K2,2	K2,3
K3,0	K3,1	K3,2	K3,3

Rijndael uses input and output at its external interface and they are one dimensional arrays of 8-bit bytes numbered upwards from 0 to the 4***N**b-1. So the length of the blocks is 16, 24 or 32 bytes. The Cipher Key has also one dimensional array which has 8 bytes numbered upwards starting from 0 to the 4*Nk-1.

3.2 The round transformation

The round transformation includes four kind of different transformations. We can write

The round transformation in pseudo code as follow:

Round (State,RoundKey)
{
ByteSub(State);
ShiftRow(State);
MixColumn(State);
AddRoundKey(State,RoundKey);
}
The final round of the cipher is slightly different. It is defined by:
FinalRound(State,RoundKey)
{
ByteSub(State) ;
ShiftRow(State) ;
AddRoundKey(State,RoundKey);
}

3.3 Key schedule

- The Round Keys are obtained from the Cipher Key using the key schedule. This can be done by two components: the key expansion and the round key selection. The main thing is that we have to consider the following rules:

- The number of Round Key bits is equal to the length of block multiplied by the number of rounds plus 1.

- The Cipher Key is expanded into an Expanded Key

4. Implementation aspects

Rijndael cipher can be implemented on many processors and in good hardware. We explain 8-bit processor. When we have 8-bit processor, we can program Rijndael implementing the different component transformations. This makes it so straightforward when it comes down to RowShift Round Key addition. If we want to implement ByteSub we need to have 256 bytes. The Round Key addition, ByteSub and RowShift can be combined and executed in the form of serial per State byte. We need to know matrix multiplication in the field of GF (2 powers 8) if we want to do the transformation MixColumn .We can show this fact as follow:

```
Tmp = a[0] ^ a[1] ^ a[2] ^ a[3] ; /* a is a byte array */
Tm = a[0] ^ a[1] ; Tm = xtime(Tm); a[0] ^= Tm ^ Tmp ;
Tm = a[1] ^ a[2] ; Tm = xtime(Tm); a[1] ^= Tm ^ Tmp ;
Tm = a[2] ^ a[3] ; Tm = xtime(Tm); a[2] ^= Tm ^ Tmp ;
Tm = a[3] ^ a[0] ; Tm = xtime(Tm); • (• ••^=Tm^Tmp;
```

Now we explain more to clarify the subject. All coding is done in assembly language. If we don't want to have timing attacks we must pay attention on xtime to implement it by taking a fixed number of cycles. We can do that by using dedicated table-lookup. We can do key expansion in a cyclic buffer of 4*max (**Nb,Nk**) bytes. The Round Key is updated in between Rounds. We can implement all operations in this key update on byte level. If the Cipher Key length and the blocks length are equal or if they differ by factor 2, the implementation is so easy. If we don't have this situation, an additional buffer pointer is required.

4.1 The inverse cipher

In the table-lookup implementation it is essential that the only non-linear step (ByteSub) is the first transformation in a round and that the rows are shifted before MixColumn is applied. In the Inverse of a round, the order of the transformations in the round is reversed, and consequently the non-linear step will end up being the last step of the inverse round and the rows are shifted after the application of (the inverse of) MixColumn. The inverse of a round can therefore not be implemented with the table lookups described above. This implementation aspect has been anticipated in the design. The structure of Rijndael is such that the sequence of transformations of its inverse is equal to that of the cipher itself, with the transformations replaced by their inverses and a change in the key schedule. This is shown in the following subsections.

The inverse of a round is given by:

```
InvRound(State,RoundKey)
{
AddRoundKey(State,RoundKey);
InvMixColumn(State);
InvShiftRow(State);
InvByteSub(State);
}
```

The inverse of the final round is given by:

```
InvFinalRound(State,RoundKey)
{
AddRoundKey(State,RoundKey);
InvShiftRow(State);
InvByteSub(State);
}
```

5. Strengths and advantages of Rijndael compare to the other known ciphers

5.1

Even though we have large amount of symmetry, this method has ways to eliminate symmetry in the behavior of the cipher. This is obtained by the round constants that are different for each round.

5.2 Differential cryptanalysis

DC attacks are possible if there are predictable difference propagation over all but a few (Typically 2 or 3) rounds that have a prop ratio (the relative amount of all input pairs that for the given input difference give rise to the output difference) significantly larger than ((2 powers (1- n)) if n is the block length.

5.3 Linear cryptanalysis

LC attacks are possible if there are predictable input-output correlation over all but a few (Typically 2 or 3) round significantly larger than (2 power n/2). An input-output correlation is composed of linear trails, where its correlation is the sum of the correlation coefficients of all linear trails that have the specified initial and final selection patterns.

5.4 Propagation of patterns

For DC, the active S-boxes in a round are determined by the nonzero bytes in the difference of the states at the input of a round. Let the pattern that specifies the positions of the active S-boxes be denoted by the term (difference) activity pattern and let the (difference) byte weight be the number of active bytes in a pattern.

5.5 Implementation

Rijndael can be implemented to run at speeds unusually fast for a block cipher on a Pentium (Pro). There is a trade-off between table size/performance. Rijndael can be implemented on a Smart Card in a small amount of code, using a small amount of RAM and taking a small number of cycles. There is some ROM/performance trade-off.

6. Weaknesses of Rijndael

There are a few weaknesses about Rijndael that most of them are related to the inverse of cipher. The most important weaknesses are as follow:

- The inverse cipher is less suited to be implemented on a smart card than the cipher itself: it takes more code and cycles.
- In software, the cipher and its inverse make use of different code and/or tables.
- In hardware, the inverse cipher can only partially re-use the circuitry that implements the cipher.

7. References

[Bi93] E. Biham, "New types of cryptanalytic attacks using related keys," Advances in Cryptology, Proceedings Eurocrypt' 93, LNCS 765, T. Helleseth, Ed., Springer-Verlag, 1993,
pp. 398-409.
[BiSh91] E. Biham and A. Shamir, "Differential cryptanalysis of DES-like cryptosystems,"
Journal of Cryptology, Vol. 4, No. 1, 1991, pp. 3-72.
[Da95] J. Daemen, "Cipher and hash function design strategies based on linear and differential
cryptanalysis," Doctoral Dissertation, March 1995, K.U.Leuven.
[DaKnRi97] J. Daemen, L.R. Knudsen and V. Rijmen, "The block cipher Square," Fast Software Encryption, LNCS 1267, E. Biham, Ed., Springer-Verlag, 1997, pp. 149-165. Also
, Geneva, 1994 (second edition).