

**Rijndael** is the new Advanced Encryption Standard.  
It was invented by Joan Daemen and Vincent Rijmen.

It is a block cipher. The block length and key length can be chosen independently to be 128, 192 or 256 bits.

It has 10, 12 or 14 rounds, depending on the block and key lengths. The rounds do not have a Feistel structure.

It was designed to be simple, to be resistant against all known attacks and to have fast and compact code on many platforms.

Mathematical preliminaries for Rijndael.

Byte operations are done with arithmetic in the field  $GF(2^8)$ .

A byte  $b_7b_6 \dots b_1b_0$  is considered a polynomial with coefficients in  $\{0, 1\}$ :

$$b(x) = b_7x^7 + b_6x^6 + \dots + b_1x + b_0.$$

Example: The byte  $0xB7 = 1011\ 0777$  is the polynomial

$$x^7 + x^5 + x^4 + x^2 + x + 1.$$

Bytes are added with XOR ( $\oplus$ ). Addition is associative and commutative. The identity element is  $0x00$ . Every byte is its own additive inverse.

Bytes are multiplied modulo  $m(x) = x^8 + x^4 + x^3 + x + 1$  ( $= 0x11B$ ). Multiplication is associative and commutative. The identity element is  $0x01$ . Every non-zero polynomial (byte) has a unique inverse with respect to this multiplication. The inverse may be computed by the extended Euclidean algorithm for GCD of the polynomial with  $m(x)$ . This multiplication is denoted  $\bullet$ .

Multiplication of  $b(x)$  by  $x = 0x02$  by a left shift of one bit position followed by a conditional XOR with  $m(x)$ : XOR with  $m(x)$  iff the bit shifted out was 1. Therefore, multiplication of two polynomials may be performed by up to 8 repeated left shifts and conditional XORs.

Thirty-two bit word operations.

Thirty-two bit words are regarded as four bytes, which are the coefficients of a polynomial of degree three with coefficients in  $GF(2^8)$ .

Addition of two 32-bit words is simple: Just add (XOR) the coefficients. This is the same as XORing the two 32-bit words.

Multiplication of two 32-bit words is done by multiplying the polynomials modulo  $M(x) = x^4 + 1$ . This multiplication is denoted  $\otimes$ . If

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$$

and

$$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0,$$

then

$$d(x) = a(x) \otimes b(x) = d_3x^3 + d_2x^2 + d_1x + d_0$$

may be computed by

$$d_0 = a_0 \bullet b_0 \oplus a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$$

$$d_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1 \oplus a_3 \bullet b_2 \oplus a_2 \bullet b_3$$

$$d_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \oplus a_3 \bullet b_3$$

$$d_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$$

Multiplication of a cubic polynomial by  $x$  consists of a circular left shift of the bytes in the word representing the polynomial.

Rijndael has 10, 12 or 14 rounds, depending on the block and key lengths. The block length and key length can be chosen independently to be 128, 192 or 256 bits. Let  $Nb$  be the length of the block in 32-bit words ( $Nb = 4, 6$  or  $8$ ). Let  $Nk$  be the length of the key in 32-bit words ( $Nk = 4, 6$  or  $8$ ). Let  $Nr$  be the number of rounds. Then  $Nr = 14$  if either  $Nb$  or  $Nk = 8$ . Otherwise,  $Nr = 12$  if either  $Nb$  or  $Nk = 6$ . Finally,  $Nr = 10$  if both  $Nb$  and  $Nk = 4$ .

Different parts of the Rijndael algorithm operate on the intermediate result, called the *State*. The State is a rectangular array of bytes with four rows and  $Nb$  columns.

The Key is expanded and placed in an array  $W[Nb*(Nr+1)]$ . The first  $Nk$  words of  $W$  are the Key. Each subsequent word is the XOR of the previous word and the word  $Nk$  words back in the array, except that words whose subscript is a multiple of  $Nk$  have the previous word transformed before the XOR.

Each round of Rijndael consists of four different transformations:

```
Round(State, RoundKey)
{
  ByteSub(State);
  ShiftRow(State);
  MixColumn(State);
  AddRoundKey(State, RoundKey);
}
```

The FinalRound omits the MixColumn.

The complete Rijndael cipher consists of:

```
Rijndael(State, CipherKey)
{
  KeyExpansion(CipherKey, ExpandedKey);
  AddRoundKey(State, ExpandedKey);
  For (i=1; i<Nr; i++) Round(State, ExpandedKey + Nb*i);
  FinalRound(State, ExpandedKey + Nb*Nr);
}
```

**ByteSub(State)** transforms each byte in the State by replacing it with its multiplicative inverse in  $GF(2^8)$  (except that **0x00** is unchanged) and then applying an affine transformation to the inverse.

**ShiftRow(State)** is a circular left shift of the rows in the State by various byte offsets which depend on **Nb**.

In **MixColumn(State)** the columns of the State are considered to be cubic polynomials with coefficients in  $GF(2^8)$  and each is multiplied ( $\otimes$ ) modulo  $x^4 + 1$  with the fixed polynomial

$$c(x) = \mathbf{0x03}x^3 + \mathbf{0x01}x^2 + \mathbf{0x01}x + \mathbf{0x02}.$$

This polynomial  $c(x)$  is relatively prime to  $x^4 + 1$  and therefore invertible. The inverse of **MixColumn(State)** is multiplication by

$$d(x) = \mathbf{0x0B}x^3 + \mathbf{0x0D}x^2 + \mathbf{0x09}x + \mathbf{0x0E}.$$

**AddRoundKey(State, RoundKey)** is simply an XOR of State with RoundKey.