Rijndael Encryption

By: Aven Bross

Galois Fields

$GF(p^n) = \{a_n p^{n-1} + \ldots + a_1 p + a_0 : a_n, \ldots, a_0 \in \mathbf{Z}_p\}$

For example $GF(2^2) = \{ 2^1 + 2^0, 2^1, 1, 0 \}$

The same as 2 digit binary: { 11, 10, 01, 00 }

Galois Arithmetic

Adding in Galois field simply adds coefficients.

$$(a_n p^{n-1} + \ldots + a_1 p + a_0) + (b_n p^{n-1} + \ldots + b_1 p + b_0)$$
$$= (a_n+b_n)p^{n-1} + \ldots + (a_1+b_1)p + (a_0+b_0)$$

Galois Arithmetic

Multiplication in Galois fields works as expected, you just distribute and then add coefficients.

For example:

$$(a_1p + a_0)\ (b_1p + b_0)$$
$$= (a_1+b_1)p^2 + [(a_1+b_0)+(a_0+b_1)]p^1 + (a_0+b_0)$$

Galois Arithmetic

However multiplication can result in a polynomial that is not in the field.

For example,

$$2^3 \in GF(2^4), \text{ but } (2^3)(2^3) = 2^6.$$

Galois Arithmetic

To remedy this, a polynomial of degree n is chosen to use as a modulus.

For our example we could choose 2^4+1.

Then, $(2^3)(2^3) = 2^6 \bmod (2^4+1) = 2^2$.

Rijndael Galois Field

Rijndael uses the Galois Field GF(2^8) and chooses the modulus polynomial

$$2^8 + 2^4 + 2^3 + 2^1 + 1.$$

Each polynomial is used to represent one byte of data in binary. For example,

$$2^6 + 2^5 + 2^3 + 2^2 + 1 = 01101101 = 109 = \text{‘m’}$$

Rijndael Data Matrices

Rijndael is a block cipher. I chose to do 128bit encryption. The data and key are each represented as a 4x4 matrix of $a(i,j) \in GF(2^8)$.

$$\begin{bmatrix} a_{(0,0)} & a_{(0,1)} & a_{(0,2)} & a_{(0,3)} \\ a_{(1,0)} & a_{(1,1)} & a_{(1,2)} & a_{(1,3)} \\ a_{(2,0)} & a_{(2,1)} & a_{(2,2)} & a_{(2,3)} \\ a_{(3,0)} & a_{(3,1)} & a_{(3,2)} & a_{(3,3)} \end{bmatrix}$$

Layer Design

Linear mixing	Shift Rows, Mix Columns
Non-linear mixing	Sub Keys
Key	Subkey Addition

Sub Bytes (S-Box)

Map each byte B in a non-linear fashion.

$A = B^{-1}$

$$A \cdot x + b = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Shift Rows

Simple operation to shift bytes around rows.

$$A = \begin{bmatrix} a_{(0,0)} & a_{(0,1)} & a_{(0,2)} & a_{(0,3)} \\ a_{(1,0)} & a_{(1,1)} & a_{(1,2)} & a_{(1,3)} \\ a_{(2,0)} & a_{(2,1)} & a_{(2,2)} & a_{(2,3)} \\ a_{(3,0)} & a_{(3,1)} & a_{(3,2)} & a_{(3,3)} \end{bmatrix} \quad A' = \begin{bmatrix} a_{(0,0)} & a_{(0,1)} & a_{(0,2)} & a_{(0,3)} \\ a_{(1,1)} & a_{(1,2)} & a_{(1,3)} & a_{(1,0)} \\ a_{(2,2)} & a_{(2,3)} & a_{(2,0)} & a_{(2,1)} \\ a_{(3,3)} & a_{(3,0)} & a_{(3,1)} & a_{(3,2)} \end{bmatrix}$$

Mix Columns

Linear mixing of columns. Transform each byte by the transformation matrix M.

$A' = M \cdot A$

$$M = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix}$$

Key Schedule

Expands 128bit key to (#rounds +1) 128bit keys. One for each round, and an initial key.

Works on “words” of 4 bytes, constructing further words by copying past bytes and applying S-Box.