Advanced Encryption Standard (AES)

In 1999, NIST decided that DES was no longer secure due to the increasing speed of computers. They solicited algorithms to replace DES to be the new government standard, AES. There were five finalists chosen, and of those, the winning entry was submitted by Joan Daemon and Vincent Rijment and is called Rijndael (pronouced "rain-doll".)

Their cipher, along with the other finalists, adhered to the following criteria laid out by NIST:

1) Symmetric block cipher with three possible key lengths: 128 bits, 192 bits and 256 bits.

2) More secure than Triple-DES

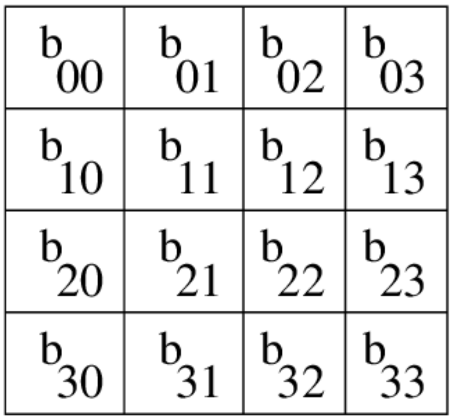
3) Must be part of the public domain, royalty free

4) It should remain secure for at least 30 years.

The algorithm utilizes mathematics with polynomials within the field GF(2^8) modulo the irreducible polynomial **$m(x) = x^8 + x^4 + x^3 + x + 1$.** The full details of this are beyond the scope of this class, but a method to carry out all the steps necessary to run AES will be given and some basic mathematical justifications will be given.

For 128-bit AES, the algorithm runs 10 rounds. For 192-bit AES, it runs 12 rounds. For 256-bit AES, it runs 14 rounds. Only the 128-bit AES will be discussed here in detail.

Assume that we have a block of 128 bits, split into 16 bytes, labeled $b_{0,0}$, $b_{1,0}$, $b_{2,0}$, $b_{3,0}$, $b_{0,1}$, $b_{1,1}$, $b_{2,1}$, $b_{3,1}$, $b_{0,2}$, $b_{1,2}$, $b_{2,2}$, $b_{3,2}$, $b_{0,3}$, $b_{1,3}$, $b_{2,3}$, and $b_{3,3}$. You can visualize these 16 bytes filling up four columns of four bytes, with the first four elements in the first column, the second four elements in the second column, etc.



b_{00}	b_{01}	b_{02}	b_{03}
b_{10}	b_{11}	b_{12}	b_{13}
b_{20}	b_{21}	b_{22}	b_{23}
b_{30}	b_{31}	b_{32}	b_{33}

The following is repeated for 10 rounds:

1) Substitute bytes

For each of the sixteen bytes, look up their substitute in the s-box substitution chart, creating the new state matrix $b_{0,0}$, $b_{1,0}$, $b_{2,0}$, $b_{3,0}$, $b_{0,1}$, $b_{1,1}$, $b_{2,1}$, $b_{3,1}$, $b_{0,2}$, $b_{1,2}$, $b_{2,2}$, $b_{3,2}$, $b_{0,3}$, $b_{1,3}$, $b_{2,3}$, and $b_{3,3}$. This s-box is actually created using mathematical inverses mod m(x) in the field GF(2^8). But, we will skip these details here. For our purposes, we can simply look up each substitution value.

2) Shift rows

In row i, perform a cyclic left shift of i bytes. (Note: The rows are numbered 0 through 3.) This will result in the following matrix:

$b_{0,0}$	$b_{0,1}$	$b_{0,2}$	$b_{0,3}$
$b_{1,1}$	$b_{1,2}$	$b_{1,3}$	$b_{1,0}$
$b_{2,2}$	$b_{2,3}$	$b_{2,0}$	$b_{2,1}$
$b_{3,3}$	$b_{3,0}$	$b_{3,1}$	$b_{3,2}$

3) Mix columns

"Multiply" the state matrix with the following matrix:

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}$$

Now this isn't regular multiplication, it's multiplication in the field mentioned before. Also, instead of adding the four terms in each operation, the four terms get XORed together. Multiplying by 01 is just the identity, and multiplying by 03 is the same as multiplying by 01 and multiplying by 02 and XORing the two results. Thus, the only thing that has to be described is multiplying by 02:

To multiply a byte by 02 (for AES, not in general), do the following:

1) Left-shift by 1 bit.
2) If the left-most bit of the original value was a 1, **get rid of this most significant bit and** XOR the remaining 8 bits from step 1 with 00011011.

Here are three examples:

4E x 02 = 01001110 x 00000010 = 10011100 = 9C (just perform the left shift)

A7 x 02 = 10100111 x 00000010 = 01001110 XOR 00011011 = 01010101 = 55

C9 x 03 = 11001001 x 01 XOR 11001001 x 02 =

```
            = 11001001 XOR 110010010
            = 11001001 XOR 10010010
                       XOR  00011011

            = 11001001 XOR 10001001

            = 11001001 XOR
              10001001
            = 01000000 (40)
```

(Note: The typical convention is to write bytes as two hex characters.)

4) Add Round Key

In this last phase of each round, the state matrix is simply XORed with the key for that particular round. In the next lecture we’ll discuss the key schedule, which produces each of the round keys.