Chapter 5

Rijndael – The Advanced Encryption Standard

5.1 History

5.1.1 Basic Facts about AES

- Successor to DES.
- The AES selection process was administered by NIST.
- Unlike DES, the AES selection was an open (i.e., public) process.
- Likely to be the dominant secret-key algorithm in the next decade.
- Main AES requirements by NIST:
  - Block cipher with 128 I/O bits
  - Three key lengths must be supported: 128/192/256 bits
  - Security relative to other submitted algorithms
  - Efficient software and hardware implementations

- See `http://www.nist.gov/aes` for further information on AES

5.1.2 Chronology of the AES Process

- Development announced on January 2, 1997 by the National Institute of Standards and Technology (NIST).
- 15 candidate algorithms accepted on August 20th, 1998.
- 5 finalists announced on August 9th, 1999
  - Mars, IBM Corporation.
  - RC6, RSA Laboratories.
  - Rijndael, J. Daemen & V. Rijmen.
  - Serpent, Eli Biham et al.
  - Twofish, B. Schneier et al.
- Monday October 2nd, 2000, NIST chooses Rijndael as the AES.

A lot of work went into software and hardware performance analysis of the AES candidate algorithms. Here are representative numbers:

Algorithm	Pentium-Pro @ 200 MHz (Mbit/sec) [WWGP00]	FPGA Hardware (Gbit/sec) [EYCP00]
MARS	69	–
RC6	105	2.4
Rijndael	71	1.9
Serpent	27	4.9
Twofish	95	1.6

Table 5.1: Speeds of the AES Finalists in Hardware and Software

5.2 Rijndael Overview

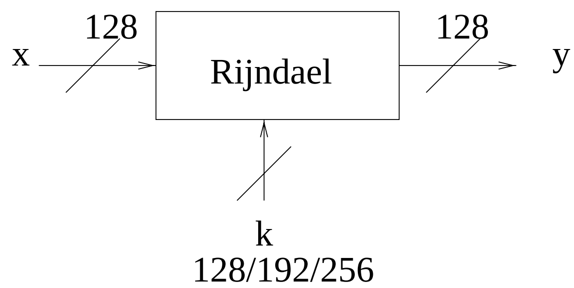

Figure 5.1: AES Block and Key Sizes

- Both blocksize and keylength of Rijndael are variable. Sizes shown in Figure 5.2 are the ones required by the AES Standard. The number of rounds (or iterations) is a function of the key length:

Key lengths (bits)	n_r = # rounds
128	10
192	12
256	14

Table 5.2: Key lenghts and number of rounds for Rijndael

- However, Rijndael also allows *blocksizes* of 192 and 256 bits. For those blocksizes the number of rounds must be increased.

Important: Rijndael does *not* have a Feistel structure. Feistel networks do not encrypt an entire block per iteration (e.g., in DES, $64/2 = 32$ bits are encrypted in one iteration). Rijndael encrypts all 128 bits in one iteration. As a consequence, Rijndael has a comparably small number of rounds.

Rijndael uses three different types of layers. Each layer operates on all 128 bits of a block:

1. *Key Addition Layer:* XORing of subkey.
2. *Byte Substitution Layer:* 8-by-8 SBox substitution.
3. *Diffusion Layer:* provides difussion over all 128 (or 192 or 256) block bits. It is split in two sub-layers:
   (a) ShiftRow Layer.
   (b) MixColumn Layer.

Remark: The ByteSubstitution Layer introduces confusion with a non-linear operation. The ShiftRow and MixColumn stages form a linear Diffusion Layer.

5.3 Some Mathematics: A Very Brief Introduction to Galois Fields

"Galois fields" are used to perform substitution and diffusion in Rijndael.

Question: What are Galois fields?

Galois fields are *fields* with a finite number of elements. Roughly speaking, a field is a structure in which we ca add, subtract, multiply, and compute inverses. More exactly a field is a ring in which all elements except 0 are invertible.

Fact 5.3.1 *Let p be a prime. $GF(p)$ is a "prime field," i.e., a Galois field with a prime number of elements. All arithmetic in $GF(p)$ is done modulo p.*

Example: $GF(3) = \{0, 1, 2\}$

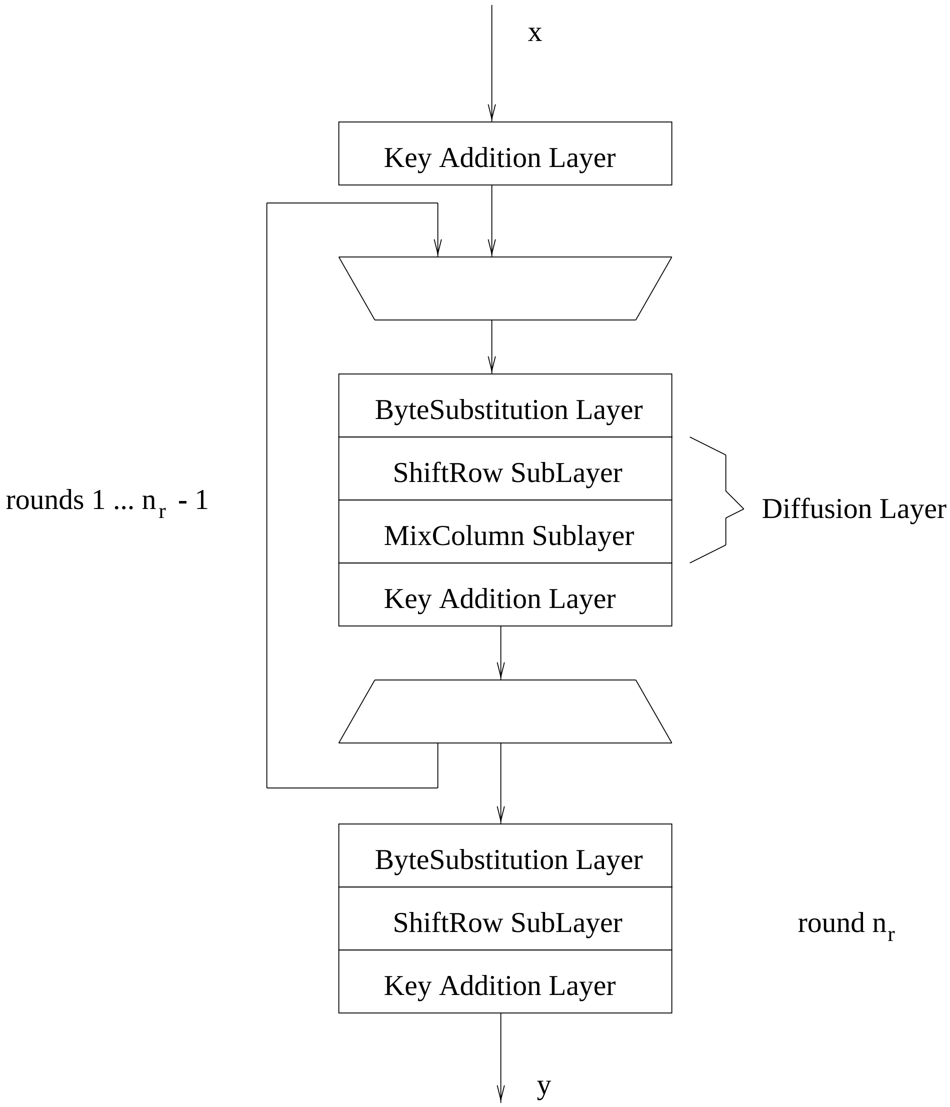

Figure 5.2: Rijndael encryption block diagram

addition

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

additive inverse

$-0 = 0$

$-1 = 2$

$-2 = 1$

multiplication

$\times$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

multiplicative inverse

0^{-1} does not exist

$1^{-1} = 1$

$2^{-1} = 2$, since $2 \cdot 2 \equiv 1 \bmod 3$

Theorem 5.3.1 *For every power p^m, p a prime and m a positive integer, there exists a finite field with p^m elements, denoted by $GF(p^m)$.*

Examples:

- $GF(5)$ is a finite field.
- $GF(256) = GF(2^8)$ is a finite field.
- $GF(12) = GF(3 \cdot 2^2)$ is **NOT** a finite field (in fact, the notation is already incorrect and you should pretend you never saw it).

Question: How to build "extension fields" $GF(p^m)$, $m > 1$?

Note: See also [Sti95, Section 5.2.1]

1. **Represent** elements as polynomials with m coefficients. Each coefficient is an element of $GF(p)$.

   Example: $A \in GF(2^8)$

   $A \rightarrow A(x) = a_7 x^7 + \cdots + a_1 x + a_0, \quad a_i \in GF(2) = \{0, 1\}$

2. **Addition and subtraction in** $GF(p^m)$

   $C(x) = A(x) + B(x) = \sum_{i=0}^{i=m-1} c_i x^i, \quad c_i = a_i + b_i \bmod p$

   Example: $A, B \in GF(2^8)$

$$\begin{array}{llllll} A(x) & = x^7 + & x^6 + & x^4 + & & 1 \\ B(x) & = & & x^4 + & x^2 + & 1 \\ \hline C(x) & = x^7 + & x^6 + & & x^2 & \end{array}$$

3. **Multiplication in $GF(p^m)$**: multiply the two polynomials using polynomial multiplication rule, with coefficient arithmetic done in $GF(p)$. The resulting polynomial will have degree $2m-2$.

$$\begin{aligned} A(x) \cdot B(x) &= (a_{m-1}x^{m-1} + \cdots + a_0) \cdot (b_{m-1}x^{m-1} + \cdots + b_0) \\ C'(x) &= c'_{2m-2}x^{2m-2} + \cdots + c'_0 \end{aligned}$$

where:

$$\begin{aligned} c'_0 &= a_0 b_0 \bmod p \\ c'_1 &= a_0 b_1 + a_1 b_0 \bmod p \\ &\vdots \\ c'_{2m-2} &= a_{m-1} b_{m-1} \bmod p \end{aligned}$$

Question: How to reduce $C'(x)$ to a polynomial of maximum degree $m-1$?

Answer: Use modular reduction, similar to multiplication in $GF(p)$. For arithmetic in $GF(p^m)$ we need an *irreducible polynomial* of degree m with coefficients from $GF(p)$. Irreducible polynomials do not factor (except trivial factor involving 1) into smaller polynomials from $GF(p)$.

Example 1: $P(x) = x^4 + x + 1$ is irreducible over $GF(2)$ and can be used to construct $GF(2^4)$.

$C = A \cdot B \Rightarrow C(x) = A(x) \cdot B(x) \bmod P(x)$

$$\begin{aligned} A(x) &= x^3 + x^2 + 1 \\ B(x) &= x^2 + x \\ C'(x) &= A(x) \cdot B(x) = (x^5 + x^4 + x^2) + (x^4 + x^3 + x) = x^5 + x^3 + x^2 + 1 \end{aligned}$$

$$
\begin{aligned}
x^4 &= 1 \cdot P(x) + (x+1) \\
x^4 &\equiv x+1 \bmod P(x) \\
x^5 &\equiv x^2 + x \bmod P(x) \\
C(x) &\equiv C'(x) \bmod P(x) \\
C(x) &\equiv (x^2+x) + (x^3+x^2+1) = x^3 \\
A(x) \cdot B(x) &\equiv x^3
\end{aligned}
$$

<u>Note</u>: in a typical computer representation, the multiplication would assign the following unusually looking operations:

$$
\begin{array}{ccccc}
A & \cdot & B & = & C \\
(1\,1\,0\,1) & \cdot & (0\,1\,1\,0) & = & (1\,0\,0\,0)
\end{array}
$$

Example 2: $x^4 + x^3 + x + 1$ is reducible since $x^4 + x^3 + x + 1 = (x^2+x+1)(x^2+1)$.

4. **Inversion in $GF(p^m)$:** the inverse A^{-1} of $A \in GF(p^m)^*$ is defined as:
   $A^{-1}(x) \cdot A(x) = 1 \bmod P(x)$
   $\Rightarrow$ perform the Extended Euclidean Algorithm with $A(x)$ and $P(x)$ as inputs

$$
\begin{aligned}
s(x)P(x) + t(x)A(x) &= \gcd(P(x), A(x)) = 1 \\
\Rightarrow t(x)A(x) &= 1 \bmod P(x) \\
\Rightarrow t(x) = A^{-1}(x) &
\end{aligned}
$$

Example: Inverse of $x^2 \in GF(2^3)$, with $P(x) = x^3 + x + 1$

$$
\begin{array}{ll}
 & t_0 = 0,\; t_1 = 1 \\
x^3 + x + 1 = [x]x^2 + [x+1] & t_2 = t_0 - q_1 t_1 = -q_1 = -x = x \\
x + 1 = [1]x + 1 & t_3 = t_1 - q_2 t_2 = 1 - q_2 x = 1 - x = x + 1 \\
x = [x]1 + 0 & \\
\Rightarrow (x^2)^{-1} = t(x) = t_3 = x + 1 &
\end{array}
$$

Check: $(x+1)x^2 = x^3 + x = (x+1) + x \equiv 1 \bmod P(x)$ since $x^3 \equiv x + 1 \bmod P(x)$.

Remark: In every iteration of the Euclidean algorithm, you should use long division (not shown above) to uniquely determine q_i and r_i.

5.4 Internal Structure

In the following, we assume a block length of 128 bits. The ShiftRow Sublayer works slightly differently for other block sizes.

5.4.1 Byte Substitution Layer

- Splits the incoming 128 bits in $128/8 = 16$ bytes.
- Each byte A is considered an element of $GF(2^8)$ and undergoes the following substitution individually
  1. $B = A^{-1} \in GF(2^8)$ where $P(x) = x^8 + x^4 + x^3 + x + 1$
  2. Apply affine transformation defined by:

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

  where $(b_7 \cdots b_0)$ is the vector representation of $B(x) = A^{-1}(x)$.

- The vector $C = (c_7 \cdots c_0)$ (representing the field element $c_7x^7 + \cdots + c_1x + c_0$) is the result of the substitution:

$$C = \text{ByteSub}(A)$$

The entire substitution can be realized as a look-up in a 256×8-bit table with fixed entries.

Remark: Unlike DES, Rijndael applies the same S-Box to each byte.

5.4.2 Diffusion Layer

- Unlike the non-linear substitution layer, the diffusion layer performs a linear operation on input words A, B. That means:

$$\text{DIFF}(A) \oplus \text{DIFF}(B) = \text{DIFF}(A + B)$$

- The diffusion layer consists of two sublayers.

ShiftRow SubLayer

1. Write an input word A as $128/8 = 16$ bytes and order them in a square array:
   Input $A = (a_0, a_1, \cdots, a_{15})$

a_0	a_4	a_8	a_{12}
a_1	a_5	a_9	a_{13}
a_2	a_6	a_{10}	a_{14}
a_3	a_7	a_{11}	a_{15}

2. Shift cyclically row-wise as follows:

$$\begin{array}{|c|c|c|c|}\hline a_0 & a_4 & a_8 & a_{12} \\ \hline a_5 & a_9 & a_{13} & a_1 \\ \hline a_{10} & a_{14} & a_2 & a_6 \\ \hline a_{15} & a_3 & a_7 & a_{11} \\ \hline \end{array} \quad \begin{array}{ll} & \text{0 positions} \\ - - - \longrightarrow & \text{3 positions right shift} \\ - - \longrightarrow & \text{2 positions right shift} \\ - \longrightarrow & \text{1 position right shift} \end{array}$$

MixColumn SubLayer

Principle: each column of 4 bytes is individually transformed into another column.

Question: How?

Each 4-byte column is considered as a vector and multiplied by a 4×4 matrix. The matirx contains *constant* entries. Multiplication and addition of the coefficients is done in $GF(2^8)$.

$$\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Remarks:

1. Each c_i, b_i is an 8-bit value representing an element from $GF(2^8)$.

2. The small values $\{01, 02, 03\}$ allow for a very efficient implementation of the coefficient multiplication in the matrix. In software implementations, multiplication by 02 and 03 can be done through table look-up in a 256-by-8 table.

3. Additions in the vector-matrix multiplication are XORs.

5.4.3 Key Addition Layer

Simple bitwise XOR with a 128-bit subkey.

5.5 Decryption

Unlike DES and other Feistel ciphers, all of Rijndael layers must actually be inverted.

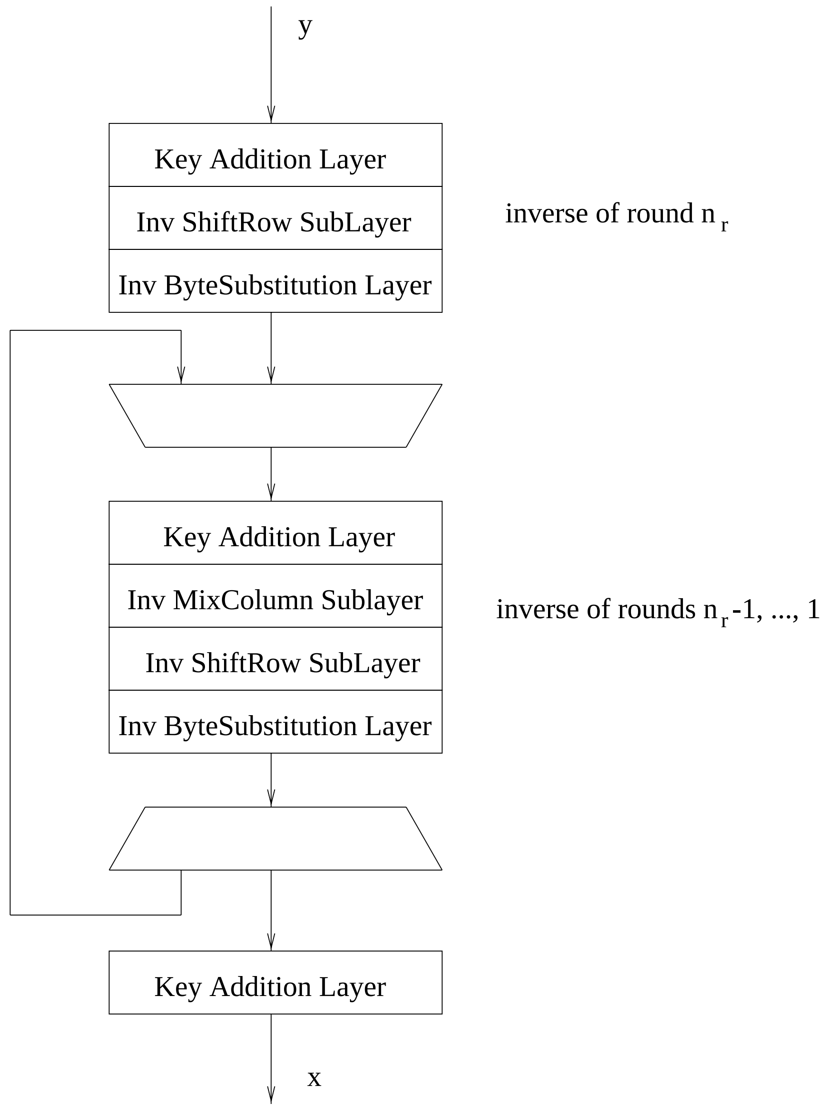

Figure 5.3: Rijndael decryption block diagram