AES (Rijndael algorithm):

1) Introduction:

This standard specifies the **Rijndael** algorithm ([3] and [4]), a symmetric block cipher that can process **data blocks** of **128 bits**, using cipher **keys** with lengths of **128**, **192**, and **256 bits**. Rijndael was designed to handle additional block sizes and key lengths, however they are not adopted in this standard.

2) Inputs and outputs:

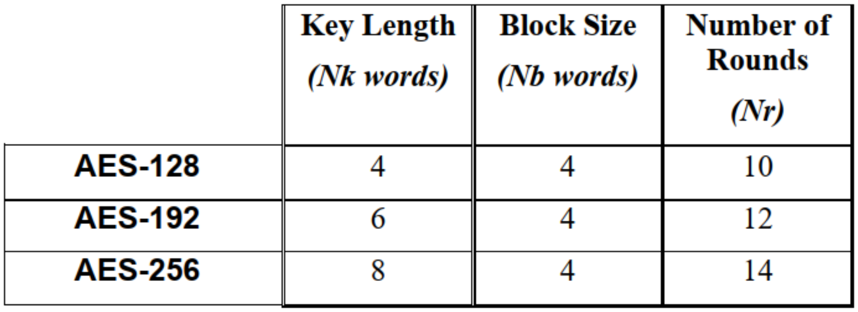
The input and output for the AES algorithm consists of sequences of 128 bits (with values 0 or 1). These sequences will sometimes be referred to as their length. The cipher key for the AES algorithm is a sequence of 128, 192 or 256 bits.

The input block length is constant and equals 128 bits, the variable is the key length. The inputs and the key are divided into 16 bytes. The basic unit for processing in the AES algorithm is a byte, a sequence of eight bits treated as a single entity.

The input, output and cipher key bit sequences described are processed as arrays of bytes that are formed by dividing these sequences into groups of eight contiguous bits to form arrays of bytes.

All byte values in the AES algorithm will be presented as the concatenation of its individual bit values (0 or 1) between braces in the order {b7, b6 ,b5, b4, b3, b2, b1 ,b0}

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0 = \sum_{i=0}^{7} b_ix^i .$$



	Key Length ***(Nk words)***	**Block Size** ***(Nb words)***	**Number of Rounds** ***(Nr)***
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Figure 1 Relation between key length and number of rounds

3) Algorithm overview:

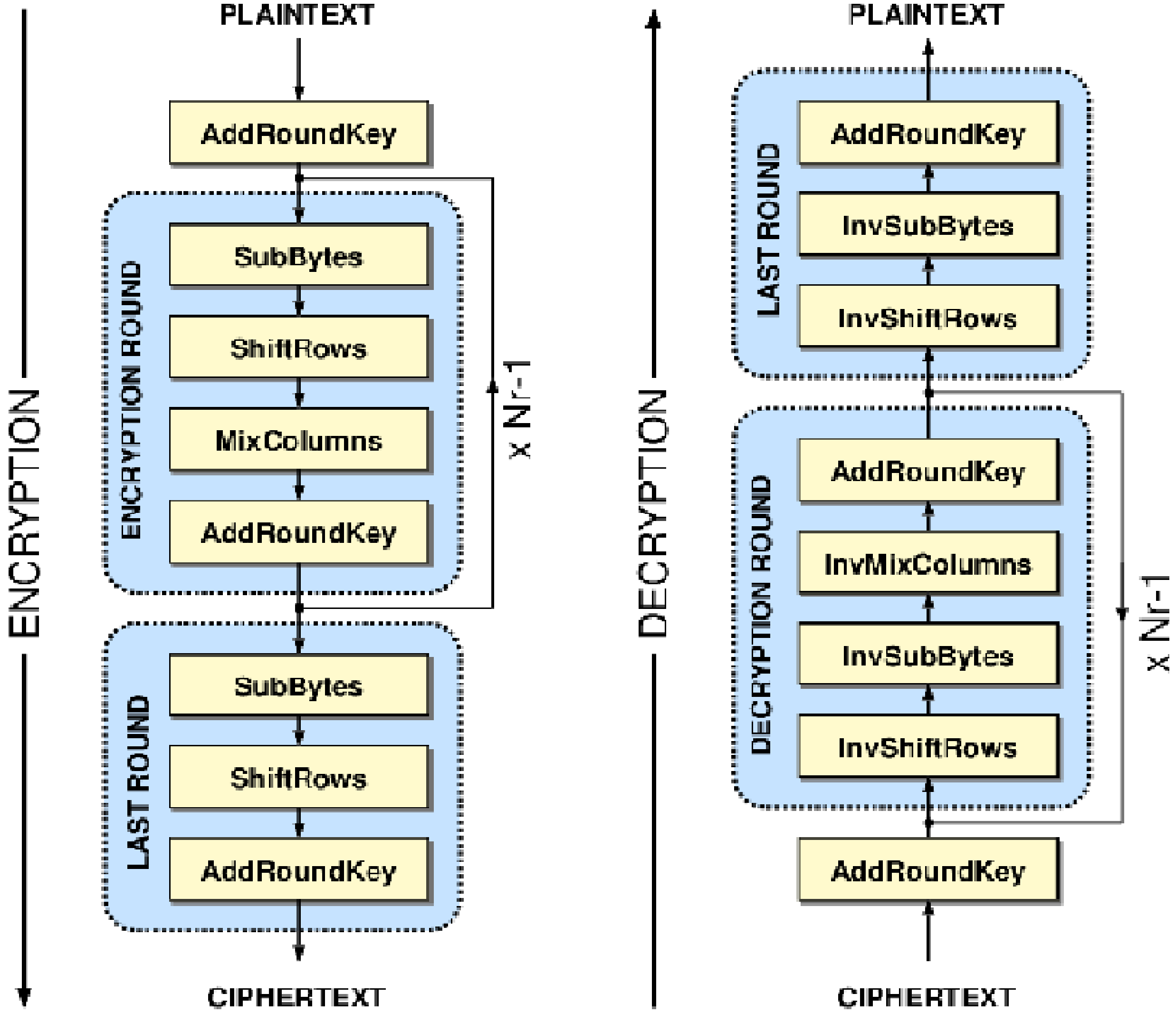


Figure 2 AES Algorithm

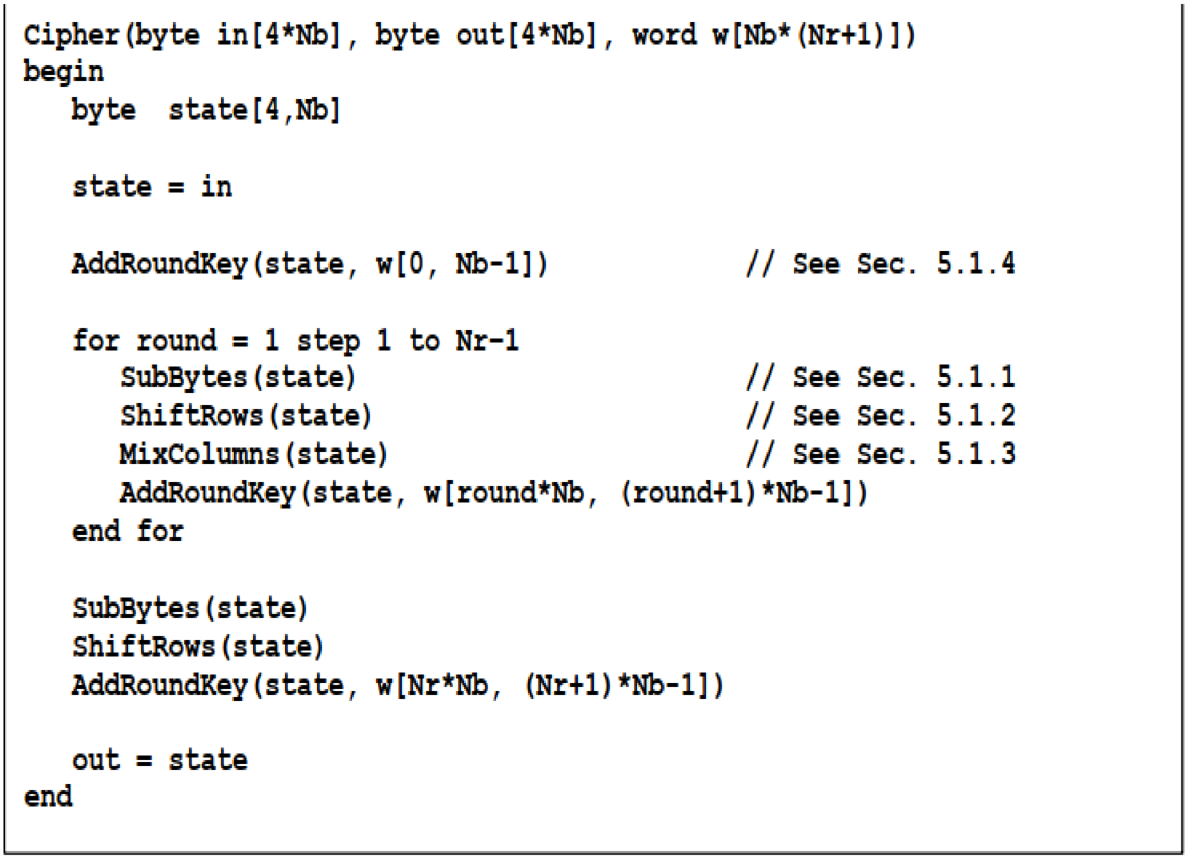
The AES algorithm consists of 4 main transformations:

- Subbytes
- Shiftrows
- Mixcolumns
- Addroundkey

The initial round adds the cipher key to the plain text using XOR addition, then we perform Nr rounds based on the length of the cipher we will use, the first Nr-1 rounds are performed using the 4 transformations, **the last round: mix columns transformation is excluded.**

During each round a key expansion is performed on the cipher key to generate a different key each round.

4) Pseudo code of Cipher:

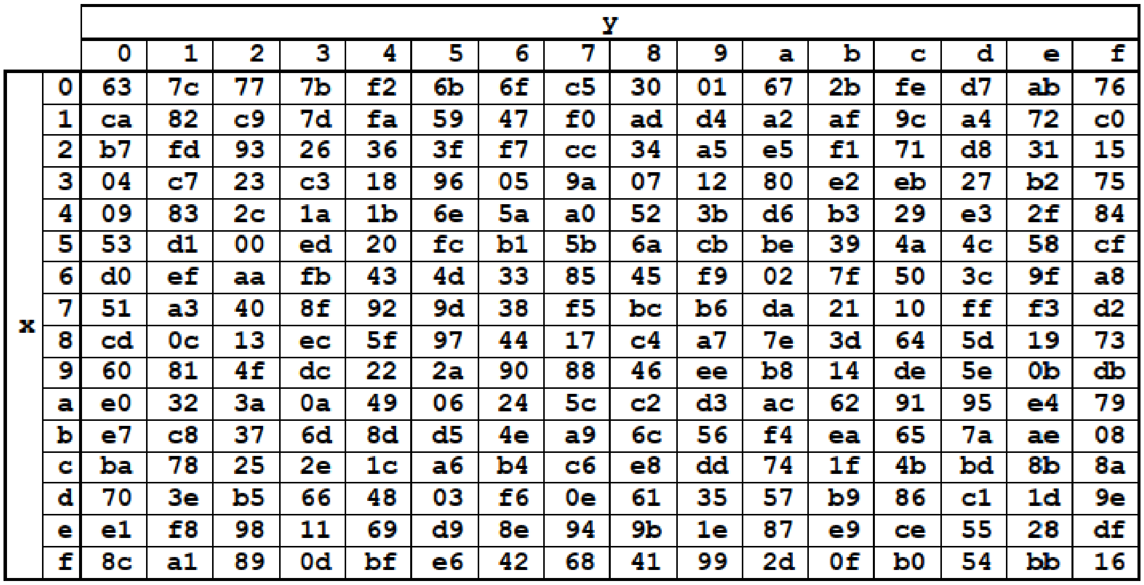


```
Cipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, w[0, Nb-1])           // See Sec. 5.1.4

   for round = 1 step 1 to Nr-1
      SubBytes(state)                       // See Sec. 5.1.1
      ShiftRows(state)                      // See Sec. 5.1.2
      MixColumns(state)                     // See Sec. 5.1.3
      AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])
   end for

   SubBytes(state)
   ShiftRows(state)
   AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])

   out = state
end
```

Figure 3 Pseudo code of AES-128 Cipher

5) Sub bytes transformation:

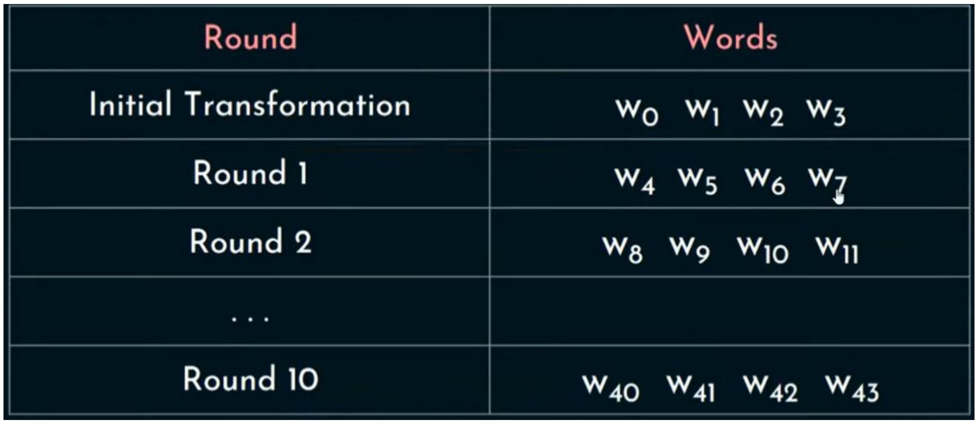
The subbytes () transformation is a non-linear byte substitution that operates independently on each byte of the state using a substitution table (S-box)



		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figure 4 S-box table

The word of the round key is 32 bits, which means that for 10 rounds, the key expansion unit exported 44 words with 4 words for each round. Now let's talk about each transformation individually.

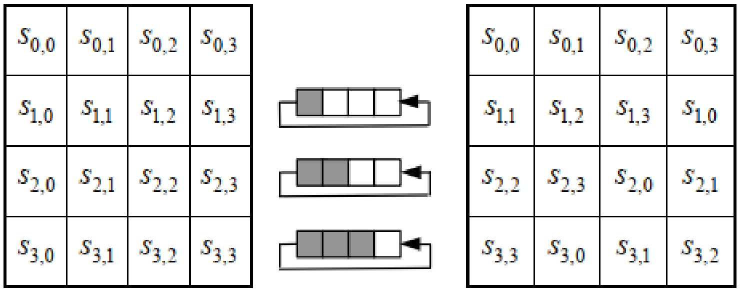


Round	Words
Initial Transformation	w_0 w_1 w_2 w_3
Round 1	w_4 w_5 w_6 w_7
Round 2	w_8 w_9 w_{10} w_{11}
. . .	
Round 10	w_{40} w_{41} w_{42} w_{43}

Figure 5 Key transformation

6) Shift Rows transformation:

In the **ShiftRows()** transformation, the bytes in the last three rows of the State are cyclically shifted over different numbers of bytes (offsets). The first row, *r* = 0, is not shifted.



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

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

Figure 6 Shift row transformation

7) Mix Columns() Transformation:

The **MixColumns()** transformation operates on the State column-by-column, treating each column as a four-term polynomial as described in Sec. 4.3. The columns are considered as polynomials over GF(28) and multiplied modulo *x*4 + 1 with a fixed polynomial *a*(*x*).

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$