Introducción a la Criptografía y a la Seguridad de la Información

Part 4
Advanced Encryption Standard

Jorge Camargo, PhD

Session 4

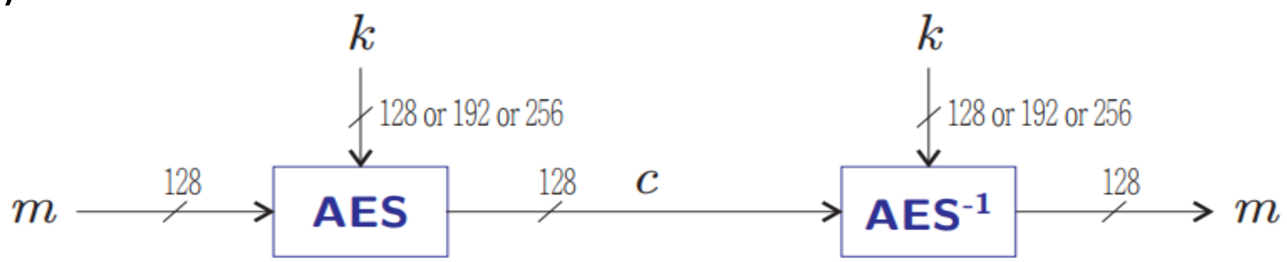
- Advanced Encryption Standard AES
  - AES Parameters
  - Data Representation
  - Steps of AES Algorithm (Encryption)
  - Steps of AES Algorithm (Decryption)
  - Key Generator
  - AddRoundKey Transformation
  - SubBytes Transformation
  - SBox Table
  - ShiftRow Transformation
  - MixColumn Transformation
  - Galois Field Multiplication
  - E-Table
  - L-Table
  - InvSubBytes Transformation
  - InvSBox Table
  - InvShiftRow Transformation
  - InvMixColumn Transformation
  - Ecryption/Decryption
  - Cipher Example
  - Decipher Example

Advanced Encryption Standard (Rijndael Cipher)

by Joan Daemen and Vincent Rijmen, 1997

The Advanced Encryption Standard (AES) is a symmetric block cipher with 128 bits block size and key sizes of 128, 192 and 256 bits.

In January 1997 the the U.S. National Institute of Standards and Technology (NIST) announced the *AES initiative* and 15 candidates were accepted for consideration. In October 2001, the highly efficient Rijndael cipher was selected as the AES cipher and the new US FIPS (Federal Information Processing Standard).



AES is currently the strongest encryption technology in the world. The U.S. government allows the use of AES-128 for sensitive and low level classified data and the AES-192 and AES-256 versions for secret and top secret data.

The name Rijndael is composed of two portions of the last names of the two Belgium authors (RIJ plus DAE).

AES Parameters

It is possible to use different key lengths (128, 192 and 256) according to the security level that is required for the application but it only defines one block length of 128 bits.

- Nb: the input/output block size in words
- Nk: the key size in words
- Nr: the number of rounds (Nr = Nk + 6)

	Parameters		
Variant	Nb	Nk	Nr
AES-128	4 words	4 words	10 rounds
AES-192	4 words	6 words	12 rounds
AES-256	4 words	8 words	14 rounds

The number of rounds to be performed during the execution of the algorithm is dependent on the key size.

A word is 32 bits.

Data Representation

The basic unit for processing in the AES algorithm is a 4×4 array of bytes, termed the state array.

First, the plain text block and the key are loaded into state arrays.

- Example: Consider the plain text “AES es muy facil”

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
A	E	S	␣	e	s	␣	m	u	y	␣	f	a	c	i	l	
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	
65	69	83	32	101	115	32	109	117	121	32	102	97	99	105	108	ASCII
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	
41	45	53	20	65	73	20	6d	75	79	20	66	61	63	69	6c	Hex

message = 41455320 6573206d 75792066 6163696c

key = 2b7e1516 28aed2a6 abf71588 09cf4f3c

state

$$\begin{pmatrix} 41 & 65 & 75 & 61 \\ 45 & 73 & 79 & 63 \\ 53 & 20 & 20 & 69 \\ 20 & 6d & 66 & 6c \end{pmatrix}$$

key

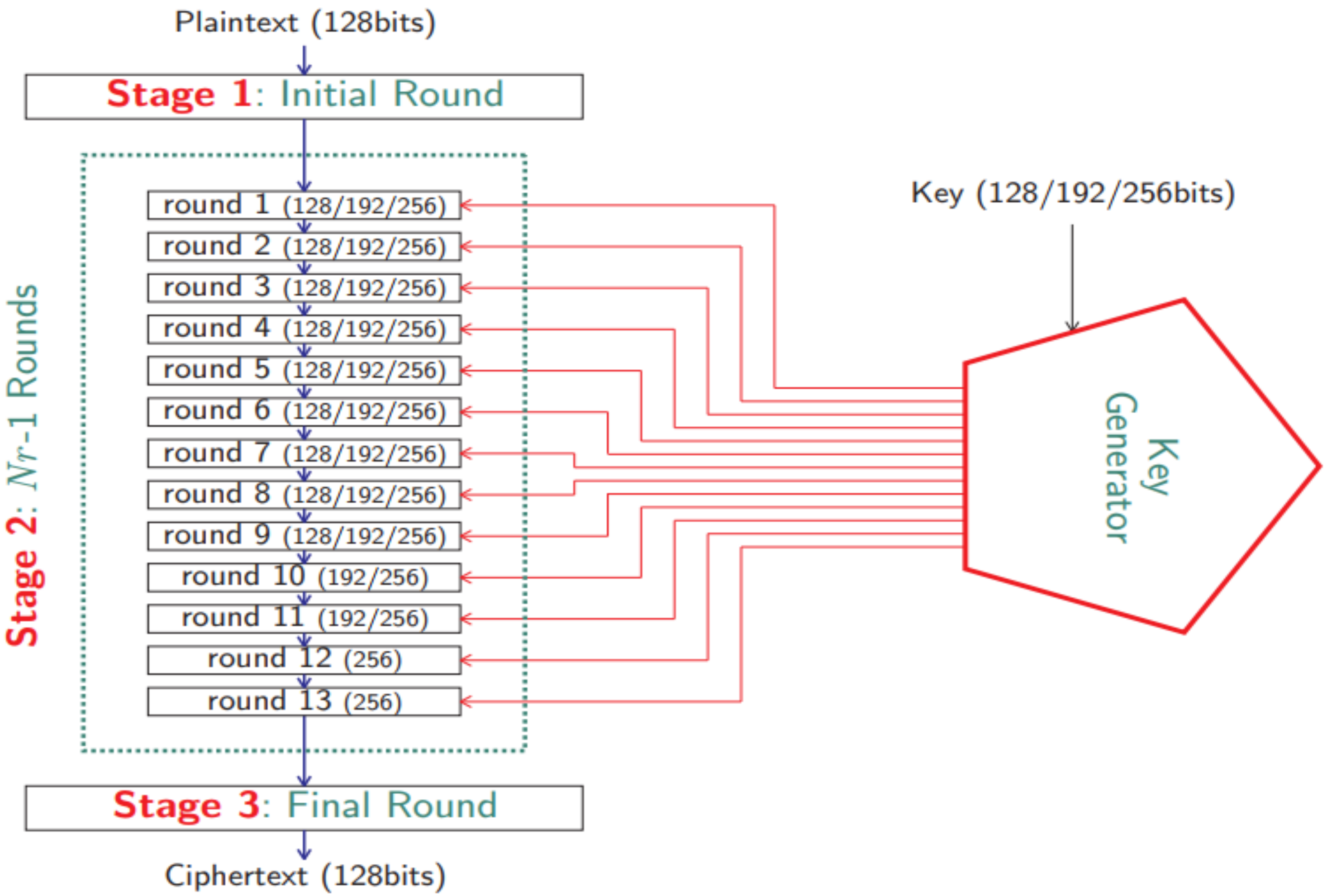
$$\begin{pmatrix} 2b & 28 & ab & 09 \\ 7e & ae & f7 & cf \\ 15 & d2 & 15 & 4f \\ 16 & a6 & 88 & 3c \end{pmatrix}$$

Steps of AES Algorithm Encryption

The algorithm has three operational stages:

- Stage 1: [Initial Round] comprising
  - AddRoundKey transformation (ARK)
- Stage 2: [Nr-1 Rounds] comprising
  - SubBytes transformation (SB)
  - ShiftRows transformation (SR)
  - MixColumns transformation (MC)
  - AddRoundKey transformation (ARK)
- Stage 3: [Final Round] comprising
  - SubBytes transformation (SB)
  - ShiftRows transformation (SR)
  - AddRoundKey transformation (ARK)

Steps of AES Algorithm Encryption (cont.)



Steps of AES Algorithm Decryption

The algorithm has three operational stages:

- Stage 1: [Initial Round] comprising
  - AddRoundKey transformation (ARK)
  - InvSubBytes transformation (SB^{-1})
  - InvShiftRows transformation (SR^{-1})
- Stage 2: [Nr-1 Rounds] comprising
  - AddRoundKey transformation (ARK)
  - InvMixColumns transformation (MC^{-1})
  - InvSubBytes transformation (SB^{-1})
  - InvShiftRows transformation (SR^{-1})
- Stage 3: [Final Round] comprising
  - AddRoundKey transformation (ARK)

Key Generator for AES-128

AES must first create Nr (10) subkeys as follows:

1. From a given key k arranged into a 4×4 matrix of bytes, we label the first four columns W[0], W[1], W[2], W[3].
2. This matrix is expanded by adding 40 more columns W[4], ⋯ , W[43] which are computed recursively as follows:

$$w[i] \begin{cases} W[i-4] \oplus T(W[i-1]), \text{ if } i \equiv 0 \pmod 4.. \\ W[i-4] \oplus W[i-1] \text{ , otherwise} \end{cases}, \text{ for } i \in [4..43]$$

   where T is the transformation of $W[i-1]$ obtained as follows: Let the elements of the column $W[i-1]$ be a, b, c, d. Shift these cyclically to obtain b, c, d, a. Now replace each of these bytes with the corresponding element in the S-Box from the ByteSub transformation to get 4 bytes e, f, g, h. Finally, compute the round constant $r[i] = 00000010(i-4)/4$ in $GF(2^8)$ then $T(W[i-1])$ is the column vector $(e \oplus r[i], f, g, h)$

3. The round key for the ith round consist of the columns W[4i], W[4i+ 1], W[4i + 2], W[4i + 3].

Key Generator for AES-128 Example

Compute all subkeys for *k* =2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c

T (W [i − 1]) = (e ⊕ r[i], f, g, h)

			1	2	3	4	3 ⊕ 4
i	$W[i\text{-}1]$	RotWord()	SubWord()	Rcon[i/4]	1 ⊕ 2	$W[i\text{-}4]$	$W[i]$
0							2b7e1516
1							28aed2a6
2							abf71588
3							09cf4f3c
4	09cf4f3c	cf4f3c09	8a84eb01	01000000	8b84eb01	2b7e1516	a0fafe17
5	a0fafe17					28aed2a6	88542cb1
6	88542cb1					abf71588	23a33939
7	23a33939					09cf4f3c	2a6c7605
8	2a6c7605	6c76052a	50386be5	02000000	52386be5	a0fafe17	f2c295f2
9	f2c295f2					88542cb1	7a96b943
10	7a96b943					23a33939	5935807a
11	5935807a					2a6c7605	7359f67f
12	7359f67f	59f67f73	cb42d28f	04000000	cf42d28f	f2c295f2	3d80477d
13	3d80477d					7a96b943	4716fe3e
14	4716fe3e					5935807a	1e237e44
15	1e237e44					7359f67f	6d7a883b

key

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

round key 1

a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

round key 2

f2	7a	59	73
c2	96	35	59
95	b9	80	f6
f2	43	7a	7f

round key 3

3d	47	1e	6d
80	16	23	7a
47	fe	7e	88
7d	3e	44	3b

Example (cont.)

			①	②	③	④	③ ⊕ ④
i	$W[i\text{-}1]$	RotWord()	SubWord()	Rcon$[i/4]$	① ⊕ ②	$W[i\text{-}4]$	$W[i]$
16	6d7a883b	7a883b6d	dac4e23c	08000000	d2c4e23c	3d80477d	ef44a541
17	ef44a541					4716fe3e	a8525b7f
18	a8525b7f					1e237e44	b671253b
19	b671253b					6d7a883b	db0bad00
20	db0bad00	0bad00db	2b9563b9	10000000	3b9563b9	ef44a541	d4d1c6f8
21	d4d1c6f8					a8525b7f	7c839d87
22	7c839d87					b671253b	caf2b8bc
23	caf2b8bc					db0bad00	11f915bc
24	11f915bc	f915bc11	99596582	20000000	b9596582	d4d1c6f8	6d88a37a
25	6d88a37a					7c839d87	110b3efd
26	110b3efd					caf2b8bc	dbf98641
27	dbf98641					11f915bc	ca0093fd
28	ca0093fd	0093fdca	63dc5474	40000000	23dc5474	6d88a37a	4e54f70e
29	4e54f70e					110b3efd	5f5fc9f3
30	5f5fc9f3					dbf98641	84a64fb2
31	84a64fb2					ca0093fd	4ea6dc4f

round key 4

ef	a8	b6	db
44	52	71	0b
a5	5b	25	ad
41	7f	3b	00

round key 5

d4	7c	ca	11
d1	83	f2	f9
c6	9d	b8	15
f8	87	bc	bc

round key 6

6d	11	db	ca
88	0b	f9	00
a3	3e	86	93
7a	fd	41	fd

round key 7

4e	5f	84	4e
54	5f	a6	a6
f7	c9	4f	dc
0e	f3	b2	4f

Example (cont.)

i	$W[i\text{-}1]$	RotWord()	SubWord() ①	Rcon[i/4] ②	① ⊕ ② ③	$W[i\text{-}4]$ ④	$W[i]$ ③ ⊕ ④
32	4ea6dc4f	a6dc4f4e	2486842f	80000000	a486842f	4e54f70e	ead27321
33	ead27321					5f5fc9f3	b58dbad2
34	b58dbad2					84a64fb2	312bf560
35	312bf560					4ea6dc4f	7f8d292f
36	7f8d292f	8d292f7f	5da515d2	1B000000	46a515d2	ead27321	ac7766f3
37	ac7766f3					b58dbad2	19fadc21
38	19fadc21					312bf560	28d12941
39	28d12941					7f8d292f	575c006e
40	575c006e	5c006e57	4a639f5b	36000000	7c639f5b	ac7766f3	d014f9a8
41	d014f9a8					19fadc21	c9ee2589
42	c9ee2589					28d12941	e13f0cc8
43	e13f0cc8					575c006e	b6630ca6

round key 8

ea	b5	31	7f
d2	8d	2b	8d
73	ba	f5	29
21	d2	60	2f

round key 9

ac	19	28	57
77	fa	d1	5c
66	dc	29	00
f3	21	41	6e

round key 10

d0	c9	e1	b6
14	ee	3f	63
f9	25	0c	0c
a8	89	c8	a6

key

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

1

a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

2

f2	7a	59	73
c2	96	35	59
95	b9	80	f6
f2	43	7a	7f

3

3d	47	1e	6d
80	16	23	7a
47	fe	7e	88
7d	3e	44	3b

4

ef	a8	b6	db
44	52	71	0b
a5	5b	25	ad
41	7f	3b	00

5

d4	7c	ca	11
d1	83	f2	f9
c6	9d	b8	15
f8	87	bc	bc

6

6d	11	db	ca
88	0b	f9	00
a3	3e	86	93
7a	fd	41	fd

7

4e	5f	84	4e
54	5f	a6	a6
f7	c9	4f	dc
0e	f3	b2	4f

8

ea	b5	31	7f
d2	8d	2b	8d
73	ba	f5	29
21	d2	60	2f

9

ac	19	28	57
77	fa	d1	5c
66	dc	29	00
f3	21	41	6e

10

d0	c9	e1	b6
14	ee	3f	63
f9	25	0c	0c
a8	89	c8	a6

AddRoundKey Transformation (ARK)

The Round Key is bitwise XORed to the State.

$$\begin{array}{rcl} 41 & = & 0100\ 0001 \\ 2b & = & \underline{0010\ 1011} \oplus \\ 6a & = & 0110\ 1010 \end{array}$$

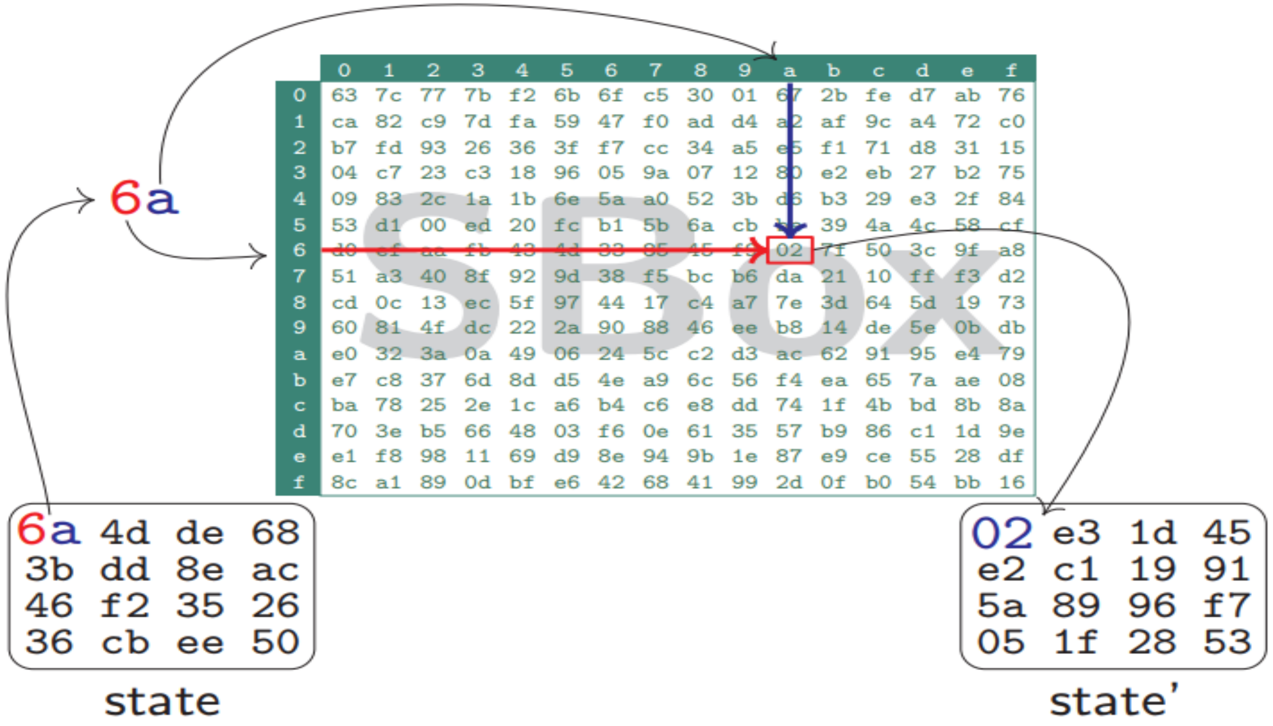
$$\begin{bmatrix} 41 & 65 & 75 & 61 \\ 45 & 73 & 79 & 63 \\ 53 & 20 & 20 & 69 \\ 20 & 6d & 66 & 6c \end{bmatrix} \oplus \begin{bmatrix} 2b & 28 & ab & 09 \\ 7e & ae & f7 & cf \\ 15 & d2 & 15 & 4f \\ 16 & a6 & 88 & 3c \end{bmatrix} = \begin{bmatrix} 6a & 4d & de & 68 \\ 3b & dd & 8e & ac \\ 46 & f2 & 35 & 26 \\ 36 & cb & ee & 50 \end{bmatrix}$$

Purpose: make the algorithm key-dependent.

Key-XORing with plaintext or ciphertext is sometimes called whitening.

SubBytes Transformation (SB)

Uses an S-Box to perform byte-by-byte substitution of the State.



	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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

Purpose: (high) non-linearity, confusion by non-linear substitution.

SBox Table

(least significant) nibble

(most significant) nibble	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
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	45	f9	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

ShiftRow Transformation (SR)

The four rows of the state array are shifted cyclically to the left as follows

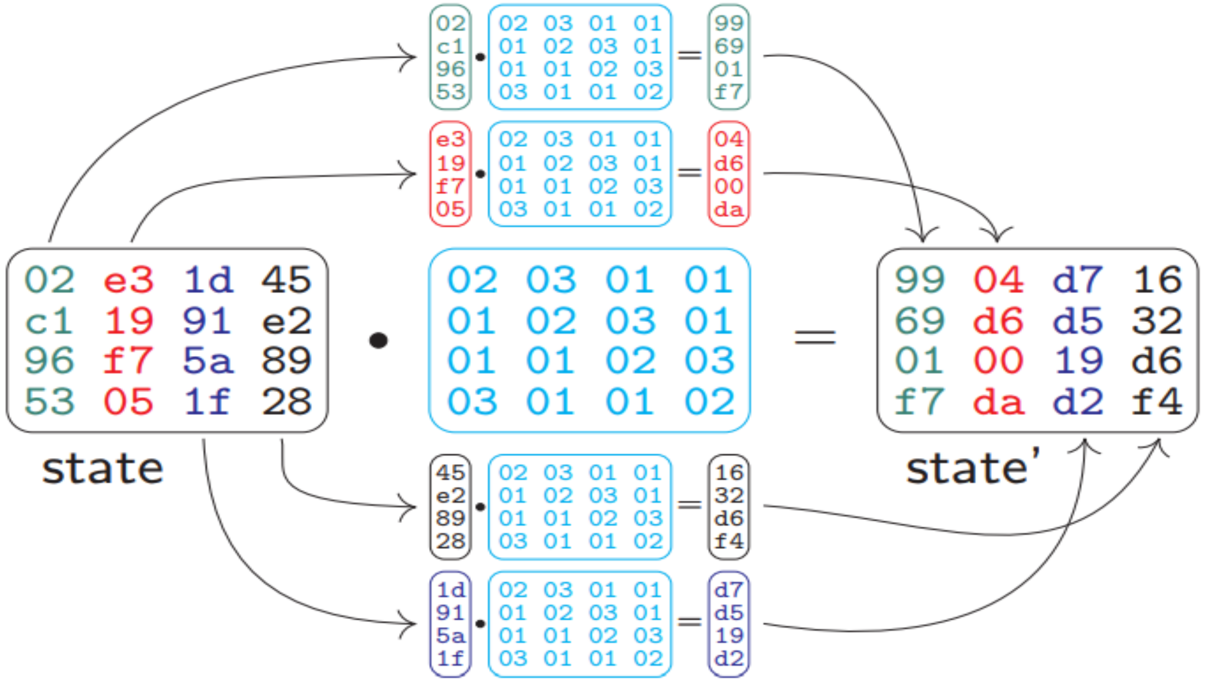
- row 0 is not shifted
- row 1 is shifted cyclically by 1 position to the left
- row 2 is shifted cyclically by 2 position to the left
- row 3 is shifted cyclically by 3 position to the left

$$\underset{\text{state}}{\boxed{\begin{array}{cccc}\mathtt{02}&\mathtt{e3}&\mathtt{1d}&\mathtt{45}\\\mathtt{e2}&\mathtt{c1}&\mathtt{19}&\mathtt{91}\\\mathtt{5a}&\mathtt{89}&\mathtt{96}&\mathtt{f7}\\\mathtt{05}&\mathtt{1f}&\mathtt{28}&\mathtt{53}\end{array}}}\begin{array}{c}0\\1\\2\\3\end{array}\rightarrow\begin{array}{ccc} & & \\ & & \mathtt{e2}\\ & \mathtt{5a} & \mathtt{89}\\\mathtt{05}&\mathtt{1f}&\mathtt{28}\end{array}\boxed{\begin{array}{cccc}\mathtt{02}&\mathtt{e3}&\mathtt{1d}&\mathtt{45}\\\mathtt{c1}&\mathtt{19}&\mathtt{91}& \\\mathtt{96}&\mathtt{f7}& & \\\mathtt{53}& & & \end{array}}=\underset{\text{state}'}{\boxed{\begin{array}{cccc}\mathtt{02}&\mathtt{e3}&\mathtt{1d}&\mathtt{45}\\\mathtt{c1}&\mathtt{19}&\mathtt{91}&\mathtt{e2}\\\mathtt{96}&\mathtt{f7}&\mathtt{5a}&\mathtt{89}\\\mathtt{53}&\mathtt{05}&\mathtt{1f}&\mathtt{28}\end{array}}}$$

Purpose: high diffusion through linear operation.

MixColumn Transformation (MC)

Each column is treated as a polynomial over GF(2^8) and is then multiplied modulo x^4+1 with a fixed polynomial 3x^3+x^2+x+2. The MixColumns transformation can also be viewed as a matrix multiply in GF(2^8).



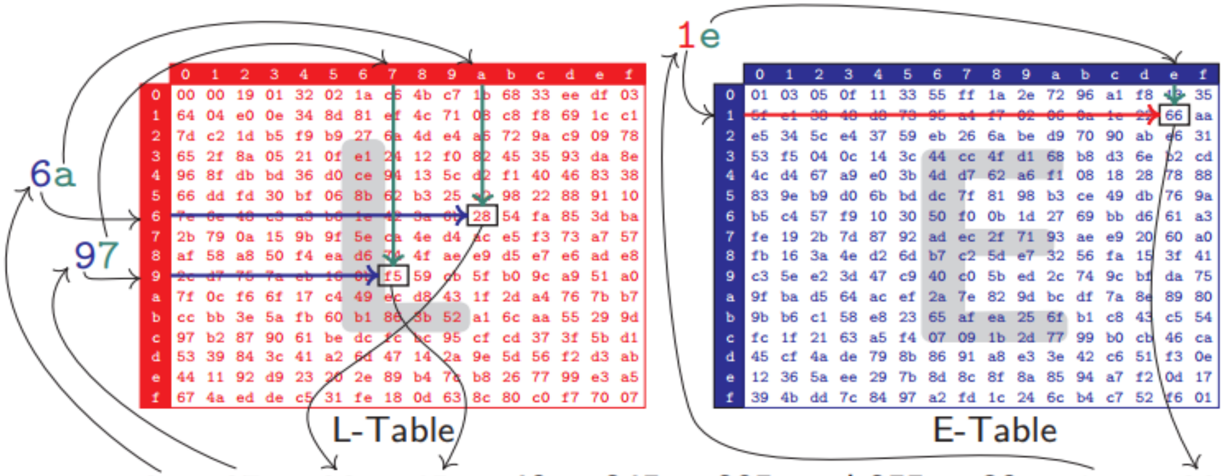
Purpose: high diffusion through linear operation.

Galois Field Multiplication

A Galois Field Multiplication can be implemented quite easily with the use of two tables: the E-Table and the L-Table.

The multiplication is simply the result of a lookup of the L-Table, followed by the addition of the results, followed by a lookup to the E-Table.

➢ **Example:** Find the multiplication of 6a and 97 in GF(2^8)



6a • 97 = 28 + f5 = 40 + 245 = 285 mod 255 = 30 = 1e = 66

40 and 245 are the decimal value of 28 and f5. 1e is the hexadecimal value of 30.

➢Example: Find the multiplication of $\begin{pmatrix} 02 \\ C1 \\ 96 \\ 53 \end{pmatrix} \cdot \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 02 & 01 & 01 & 02 \end{pmatrix}$ in GF(2^8)

$$02 \bullet 01 \oplus c1 \bullet 02 \oplus 96 \bullet 03 \oplus 53 \bullet 01 = 02 \oplus 99 \oplus a1 \oplus 53 = 69$$

$$02 \bullet 03 \oplus c1 \bullet 01 \oplus 96 \bullet 01 \oplus 53 \bullet 02 = 06 \oplus c1 \oplus 96 \oplus a6 = f7$$

$$\begin{pmatrix} 02 \\ c1 \\ 96 \\ 53 \end{pmatrix} \bullet \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} = \begin{pmatrix} 99 \\ 69 \\ 01 \\ f7 \end{pmatrix}$$

$$02 \bullet 02 \oplus c1 \bullet 03 \oplus 96 \bullet 01 \oplus 53 \bullet 01 = 04 \oplus 58 \oplus 96 \oplus 53 = 99$$

$$02 \bullet 01 \oplus c1 \bullet 01 \oplus 96 \bullet 02 \oplus 53 \bullet 03 = 02 \oplus c1 \oplus 37 \oplus f5 = 01$$

E-Table

(least significant) nibble

(most significant) nibble

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

L-Table

(least significant) nibble

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

InvSubBytes Transformation (SB^{-1})

The InvSubBytes Transformation is another lookup table using table InvSBox.

InvShiftRow Transformation (SR^{-1})

The inverse of ShiftRow is obtained by shifting the rows to the right instead of the left.

InvMixColumn Transformation (MC^{-1})

The inverse of MixColumn exists because the 4×4 matrix used in MixColumn is invetible. The transformation InvMixColumn is given by multiplying by the following matrix.

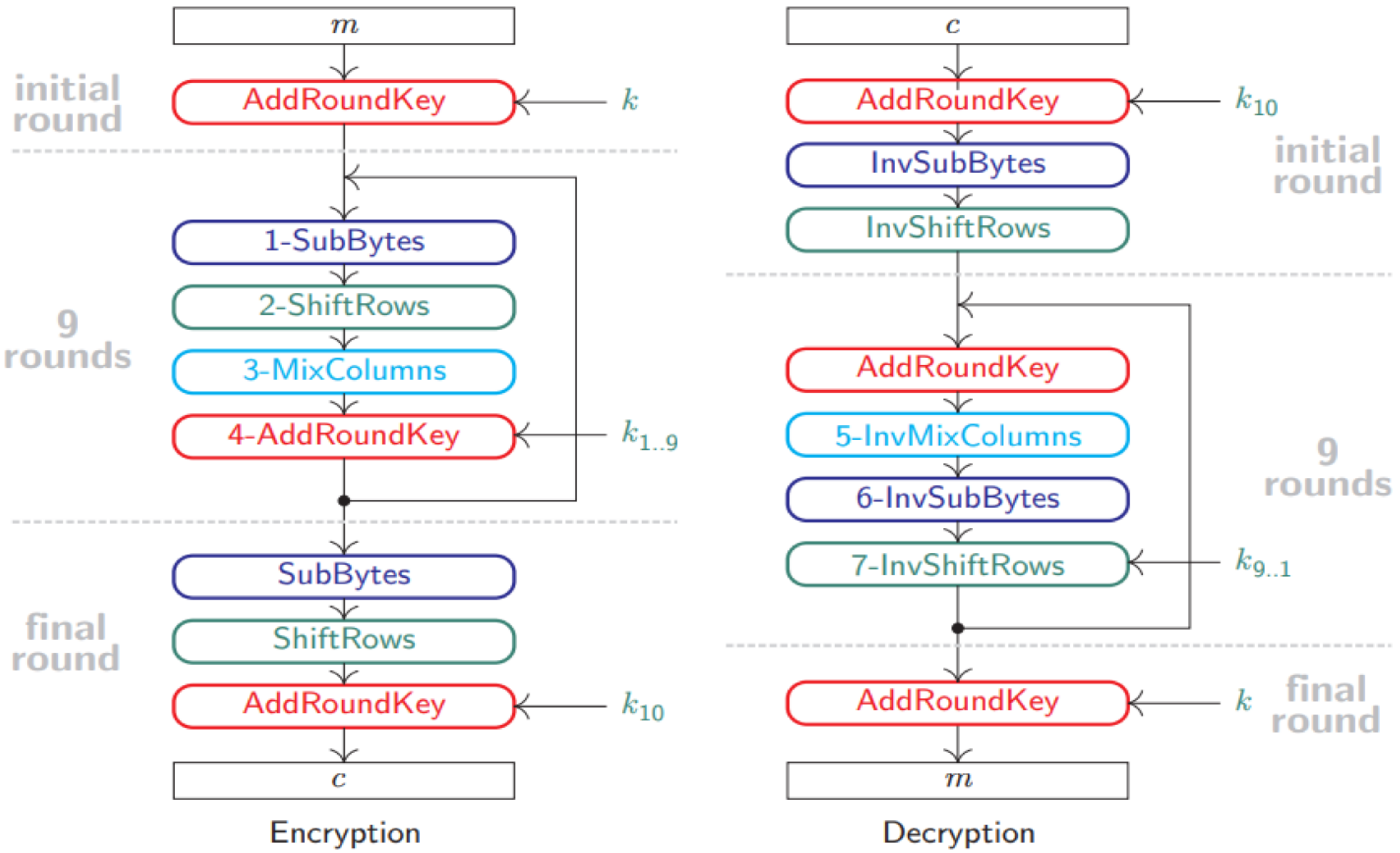
$$\begin{pmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{pmatrix}$$

InvSBox Table

(least significant) nibble

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

Encryption/Decryption



Cipher Example

Let m = 41 45 53 20 65 73 20 6d 75 79 20 66 61 63 69 6c and k = 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c, where m and k are in hexadecimal (base 16) format.

Part 1: Create 10 subkeys: as shown before, we have

key

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

subkey 1

a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

subkey 2

f2	7a	59	73
c2	96	35	59
95	b9	80	f6
f2	43	7a	7f

subkey 3

3d	47	1e	6d
80	16	23	7a
47	fe	7e	88
7d	3e	44	3b

subkey 4

ef	a8	b6	db
44	52	71	0b
a5	5b	25	ad
41	7f	3b	00

subkey 5

d4	7c	ca	11
d1	83	f2	f9
c6	9d	b8	15
f8	87	bc	bc

subkey 6

6d	11	db	ca
88	0b	f9	00
a3	3e	86	93
7a	fd	41	fd

subkey 7

4e	5f	84	4e
54	5f	a6	a6
f7	c9	4f	dc
0e	f3	b2	4f

subkey 8

ea	b5	31	7f
d2	8d	2b	8d
73	ba	f5	29
21	d2	60	2f

subkey 9

ac	19	28	57
77	fa	d1	5c
66	dc	29	00
f3	21	41	6e

subkey 10

d0	c9	e1	b6
14	ee	3f	63
f9	25	0c	0c
a8	89	c8	a6

Cipher Example (cont.)

Part 2: Encode each 128-bit block of data.

round	① ARK(④,⑤)	② SB(①)	③ SR(②)	④ MC(③)	⑤ round key
input	41 65 75 61 45 73 79 63 53 20 20 69 20 6d 66 6c				2b 28 ab 09 7e ae f7 cf 15 d2 15 4f 16 a6 88 3c
1	6a 4d de 68 3b dd 8e ac 46 f2 35 26 36 cb ee 50	02 e3 1d 45 e2 c1 19 91 5a 89 96 f7 05 1f 28 53	02 e3 1d 45 c1 19 91 e2 96 f7 5a 89 53 05 1f 28	99 04 d7 16 69 d6 d5 32 01 00 19 d6 f7 da d2 f4	a0 88 23 2a fa 54 a3 6c fe 2c 39 76 17 b1 39 05
2	39 8c f4 3c 93 82 76 5e ff 2c 20 a0 e0 6b eb f1	12 64 bf eb dc 13 38 58 16 71 b7 e0 e1 7f e9 a1	12 64 bf eb 13 38 58 dc b7 e0 16 71 a1 e1 7f e9	07 81 e4 2a 57 ce 4a 32 8c bf 4a f5 cb ad 6a 42	f2 7a 59 73 c2 96 35 59 95 b9 80 f6 f2 43 7a 7f
3	f5 fb bd 59 95 58 7f 6b 19 06 ca 03 39 ee 10 3d	e6 0f 7a cb 2a 6a d2 7f d4 6f 74 7b 12 28 ca 27	e6 0f 7a cb 6a d2 7f 2a 74 7b d4 6f 27 12 28 ca	3a 1a 89 56 89 2f cb e4 0d 1d ce 7a 61 9c 75 8c	3d 47 1e 6d 80 16 23 7a 47 fe 7e 88 7d 3e 44 3b

Cipher Example (cont.)

round	ARK(④,⑤) ①	SB(①) ②	SR(②) ③	MC(③) ④	round key ⑤
4	07 5d 97 3b 09 39 e8 9e 4a e3 b0 f2 1c a2 31 b7	c5 4c 88 e2 01 12 9b 0b d6 11 e7 89 9c 3a c7 a9	c5 4c 88 e2 12 9b 0b 01 e7 89 d6 11 a9 9c 3a c7	e9 3b fa 0a 7a 7d c5 14 e2 61 7a 93 e8 e5 2a b8	ef a8 b6 db 44 52 71 0b a5 5b 25 ad 41 7f 3b 00
5	06 93 4c d1 3e 2f b4 1f 47 3a 5f 3e a9 9a 11 b8	6f dc 29 3e b2 15 8d c0 a0 80 cf b2 d3 b8 82 6c	6f dc 29 3e 15 8d c0 b2 cf b2 a0 80 6c d3 b8 82	42 4e 11 b3 63 c3 f1 58 4b 40 61 0a b3 fd 70 6f	d4 7c ca 11 d1 83 f2 f9 c6 9d b8 15 f8 87 bc bc
6	96 32 db a2 b2 40 03 a1 8d dd d9 1f 4b 7a cc d3	90 23 b9 3a 37 09 7b 32 5d c1 35 c0 b3 da 4b 66	90 23 b9 3a 09 7b 32 37 35 c0 5d c1 66 b3 da 4b	73 b8 b8 a7 bb 3d e0 47 59 0d 44 49 5b a3 10 2e	6d 11 db ca 88 0b f9 00 a3 3e 86 93 7a fd 41 fd
7	1e a9 63 6d 33 36 19 47 fa 33 c2 da 21 5e 51 d3	72 d3 fb 3c c3 05 d4 a0 2d c3 25 57 fd 58 d1 66	72 d3 fb 3c 05 d4 a0 c3 25 57 2d c3 66 fd 58 d1	a8 70 63 34 71 64 8f 2e 97 b5 e9 0a 7a 0c 2b fd	4e 5f 84 4e 54 5f a6 a6 f7 c9 4f dc 0e f3 b2 4f

Cipher Example (cont.)

round	① ARK(④,⑤)	② SB(①)	③ SR(②)	④ MC(③)	⑤ round key
8	e6 2f e7 7a 25 3b 29 88 60 7c a6 d6 74 ff 99 b2	8e 15 94 da 3f e2 a5 c4 d0 10 24 f6 92 16 ee 37	8e 15 94 da e2 a5 c4 3f 24 f6 d0 10 37 92 16 ee	29 ba a2 10 0a d7 7a 7a 7d ea d1 ec 21 53 9f 9d	ea b5 31 7f d2 8d 2b 8d 73 ba f5 29 21 d2 60 2f
9	c3 0f 93 6f d8 5a 51 f7 0e 50 24 c5 00 81 ff b2	2e 76 dc a8 61 be d1 68 ab 53 36 a6 63 0c 16 37	2e 76 dc a8 be d1 68 61 36 a6 ab 53 37 63 0c 16	84 41 bc ad 24 5d e6 89 a5 55 ed 55 94 2b a4 fd	ac 19 28 57 77 fa d1 5c 66 dc 29 00 f3 21 41 6e
10	28 58 94 fa 53 a7 37 d5 c3 89 c4 55 67 0a e5 93	34 6a 22 2d ed 5c 9a 03 2e a7 1c fc 85 67 d9 dc	34 6a 22 2d 5c 9a 03 ed 1c fc 2e a7 dc 85 67 d9		d0 c9 e1 b6 14 ee 3f 63 f9 25 0c 0c a8 89 c8 a6
output	e4 a3 c3 9b 48 74 3c 8e e5 d9 22 ab 74 0c af 7f				

Therefore, the encrypted form of m = 41 45 53 20 65 73 20 6d 75 79 20 66 61 63 69 6c is c = e4 48 e5 74 a3 74 d9 0c c3 3c 22 af 9b 8e ab 7f.

Decipher Example

Decrypt c = e4 48 e5 74 a3 74 d9 0c c3 3c 22 af 9b 8e ab 7f using k = 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c as key.

	①	②	③	④	⑤
round	ARK(④,⑤)	MC^{-1}(①)	SR^{-1}(②)	SB^{-1}(③)	round key
input	e4 a3 c3 9b 48 74 3c 8e e5 d9 22 ab 74 0c af 7f				d0 c9 e1 b6 14 ee 3f 63 f9 25 0c 0c a8 89 c8 a6
10	34 6a 22 2d 5c 9a 03 ed 1c fc 2e a7 dc 85 67 d9		34 6a 22 2d ed 5c 9a 03 2e a7 1c fc 85 67 d9 dc	28 58 94 fa 53 a7 37 d5 c3 89 c4 55 67 0a e5 93	ac 19 28 57 77 fa d1 5c 66 dc 29 00 f3 21 41 6e
9	84 41 bc ad 24 5d e6 89 a5 55 ed 55 94 2b a4 fd	2e 76 dc a8 be d1 68 61 36 a6 ab 53 37 63 0c 16	2e 76 dc a8 61 be d1 68 ab 53 36 a6 63 0c 16 37	c3 0f 93 6f d8 5a 51 f7 0e 50 24 c5 00 81 ff b2	ea b5 31 7f d2 8d 2b 8d 73 ba f5 29 21 d2 60 2f
8	29 ba a2 10 0a d7 7a 7a 7d ea d1 ec 21 53 9f 9d	8e 15 94 da e2 a5 c4 3f 24 f6 d0 10 37 92 16 ee	8e 15 94 da 3f e2 a5 c4 d0 10 24 f6 92 16 ee 37	e6 2f e7 7a 25 3b 29 88 60 7c a6 d6 74 ff 99 b2	4e 5f 84 4e 54 5f a6 a6 f7 c9 4f dc 0e f3 b2 4f

Decipher Example (cont.)

	①	②	③	④	⑤
round	ARK(④,⑤)	MC^{-1}(①)	SR^{-1}(②)	SB^{-1}(③)	round key
7	a8 70 63 34 71 64 8f 2e 97 b5 e9 0a 7a 0c 2b fd	72 d3 fb 3c 05 d4 a0 c3 25 57 2d c3 66 fd 58 d1	72 d3 fb 3c c3 05 d4 a0 2d c3 25 57 fd 58 d1 66	1e a9 63 6d 33 36 19 47 fa 33 c2 da 21 5e 51 d3	6d 11 db ca 88 0b f9 00 a3 3e 86 93 7a fd 41 fd
6	73 b8 b8 a7 bb 3d e0 47 59 0d 44 49 5b a3 10 2e	90 23 b9 3a 09 7b 32 37 35 c0 5d c1 66 b3 da 4b	90 23 b9 3a 37 09 7b 32 5d c1 35 c0 b3 da 4b 66	96 32 db a2 b2 40 03 a1 8d dd d9 1f 4b 7a cc d3	d4 7c ca 11 d1 83 f2 f9 c6 9d b8 15 f8 87 bc bc
5	42 4e 11 b3 63 c3 f1 58 4b 40 61 0a b3 fd 70 6f	6f dc 29 3e 15 8d c0 b2 cf b2 a0 80 6c d3 b8 82	6f dc 29 3e b2 15 8d c0 a0 80 cf b2 d3 b8 82 6c	06 93 4c d1 3e 2f b4 1f 47 3a 5f 3e a9 9a 11 b8	ef a8 b6 db 44 52 71 0b a5 5b 25 ad 41 7f 3b 00
4	e9 3b fa 0a 7a 7d c5 14 e2 61 7a 93 e8 e5 2a b8	c5 4c 88 e2 12 9b 0b 01 e7 89 d6 11 a9 9c 3a c7	c5 4c 88 e2 01 12 9b 0b d6 11 e7 89 9c 3a c7 a9	07 5d 97 3b 09 39 e8 9e 4a e3 b0 f2 1c a2 31 b7	3d 47 1e 6d 80 16 23 7a 47 fe 7e 88 7d 3e 44 3b

Decipher Example (cont.)

	①	②	③	④	⑤
round	ARK(④,⑤)	MC^{-1}(①)	SR^{-1}(②)	SB^{-1}(③)	round key
3	3a 1a 89 56 89 2f cb e4 0d 1d ce 7a 61 9c 75 8c	e6 0f 7a cb 6a d2 7f 2a 74 7b d4 6f 27 12 28 ca	e6 0f 7a cb 2a 6a d2 7f d4 6f 74 7b 12 28 ca 27	f5 fb bd 59 95 58 7f 6b 19 06 ca 03 39 ee 10 3d	f2 7a 59 73 c2 96 35 59 95 b9 80 f6 f2 43 7a 7f
2	07 81 e4 2a 57 ce 4a 32 8c bf 4a f5 cb ad 6a 42	12 64 bf eb 13 38 58 dc b7 e0 16 71 a1 e1 7f e9	12 64 bf eb dc 13 38 58 16 71 b7 e0 e1 7f e9 a1	39 8c f4 3c 93 82 76 5e ff 2c 20 a0 e0 6b eb f1	a0 88 23 2a fa 54 a3 6c fe 2c 39 76 17 b1 39 05
1	99 04 d7 16 69 d6 d5 32 01 00 19 d6 f7 da d2 f4	02 e3 1d 45 c1 19 91 e2 96 f7 5a 89 53 05 1f 28	02 e3 1d 45 e2 c1 19 91 5a 89 96 f7 05 1f 28 53	6a 4d de 68 3b dd 8e ac 46 f2 35 26 36 cb ee 50	2b 28 ab 09 7e ae f7 cf 15 d2 15 4f 16 a6 88 3c
output	41 65 75 61 45 73 79 63 53 20 20 69 20 6d 66 6c				

Therefore, the decrypted message is m = 41 45 53 20 65 73 20 6d 75 79 20 66 61 63 69 6c corresponding to the message “AES es muy facil”.

References

- Pinzon, Yoan. “Introducción a la criptografía y a la seguridad de la información”, 2013.
- Menezes A., Handbook of applied Cryptografy, 5th Edition. CRC Press, 2001.
- A Graduate Course in Applied Cryptography by D. Boneh and V. Shoup
- H. Delfs and H. Knebl, Introduction to Cryptography, 3rd ed. 2015.
- J. A. Buchmann, Introduction to Cryptography. 2004.
- T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, Ch. 31 in Introduction to Algorithms, 3rd ed. 2009.
- B. Lomas de Zamora: Gradi. Hacking desde cero, Fox Andina, 2011.
- Secure Coding Working Group, Japan Smartphone Security Association (JSSEC), Android Application Secure Design/Secure Coding Guidebook, 2017.