République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Ibn Khaldoun - Tiaret-

Faculté des mathématiques et de l'informatique

TP4: AES 128

Les noms des étudiantes :

Mousselmal Sarra Groupe : 01 GL

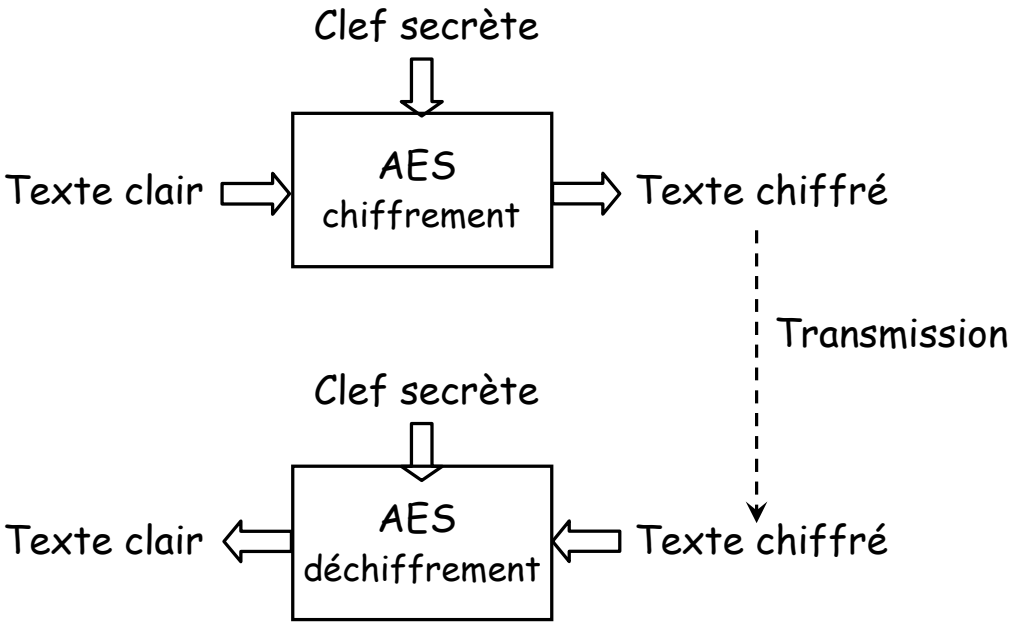
L'enseignant :Laouni DJAFRI.

Année scolaire : 2022/2023

AES128bits.

Algorithme de chiffrement/déchiffrement symétrique (i.e.à clef secrète).

→transmission d'un message confidentiel via un canal non sécurisé.



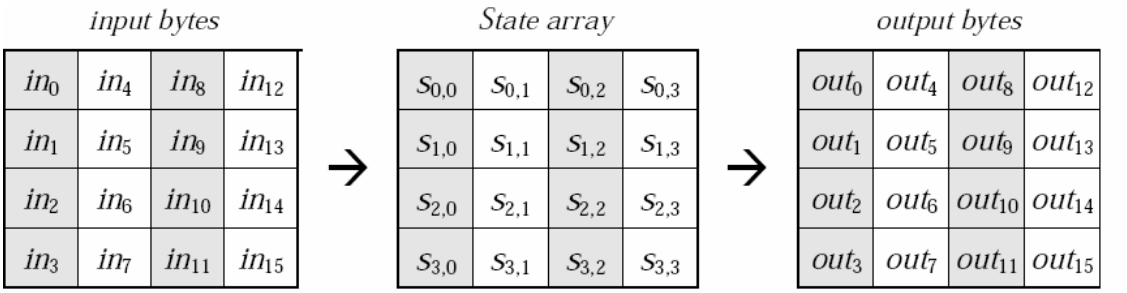
Une même clef secrète est utilisée pour les opérations de chiffrement et de déchiffrement(c'est un secretpartagé[1]entrel'expéditeur et le destinataire du message).

AES est un algorithme de chiffrement par blocs, les données sont traitées par blocs de 128bits pour le texte clair et le chiffré. La clef secrète a une longueur de 128 bits, d'où le nom de version : AES 128 (il existe deux autres variantes dont la clef fait respectivement 192 et 256bits).

Conventions , indices , typage.

Convention de notation et indices de la norme:

Input bit sequence	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	…
Byte number	0								1								2								…



s[r,c] =in[r+4c] out[r + 4c] =

s[r,c]avec r *row number*

$0 \leq r < 4$

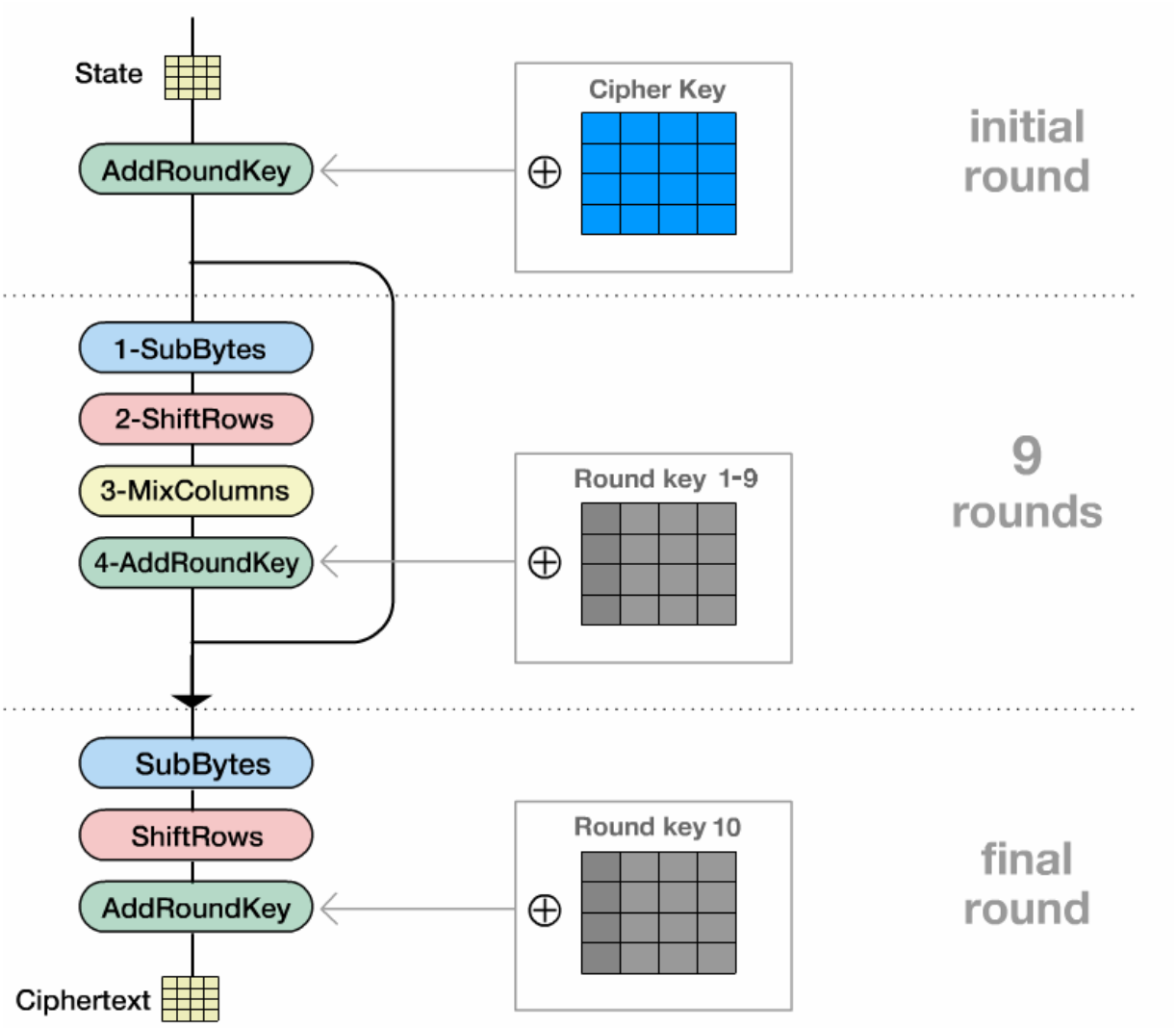
c *column number* $0 \leq c < 4$

Un état peut également être considéré comme un table au de colonnes, chaque colonne comprenant 4 octets / 32 bits, i.e. de mots (*word*) de32 bits.

$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$
w_0	w_1	w_2	w_3

$$w_c = \begin{matrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{matrix}$$

Chiffrement–*Cipher*.



La première chose que nous devons faire est de convertir le texte que nous voulons chiffrer en hexadécimal avec cette table

Binary	Oct	Dec	Hex	Glyph
010 0000	040	32	20	
010 0001	041	33	21	!
010 0010	042	34	22	"
010 0011	043	35	23	#
010 0100	044	36	24	$
010 0101	045	37	25	%
010 0110	046	38	26	&
010 0111	047	39	27	'
010 1000	050	40	28	(
010 1001	051	41	29	)
010 1010	052	42	2A	*
010 1011	053	43	2B	+
010 1100	054	44	2C	,
010 1101	055	45	2D	-
010 1110	056	46	2E	.
010 1111	057	47	2F	/

Binary	Oct	Dec	Hex	Glyph
100 0000	100	64	40	@
100 0001	101	65	41	A
100 0010	102	66	42	B
100 0011	103	67	43	C
100 0100	104	68	44	D
100 0101	105	69	45	E
100 0110	106	70	46	F
100 0111	107	71	47	G
100 1000	110	72	48	H
100 1001	111	73	49	I
100 1010	112	74	4A	J
100 1011	113	75	4B	K
100 1100	114	76	4C	L
100 1101	115	77	4D	M
100 1110	116	78	4E	N
100 1111	117	79	4F	O

Binary	Oct	Dec	Hex	Glyph
110 0000	140	96	60	`
110 0001	141	97	61	a
110 0010	142	98	62	b
110 0011	143	99	63	c
110 0100	144	100	64	d
110 0101	145	101	65	e
110 0110	146	102	66	f
110 0111	147	103	67	g
110 1000	150	104	68	h
110 1001	151	105	69	i
110 1010	152	106	6A	j
110 1011	153	107	6B	k
110 1100	154	108	6C	l
110 1101	155	109	6D	m
110 1110	156	110	6E	n
110 1111	157	111	6F	o

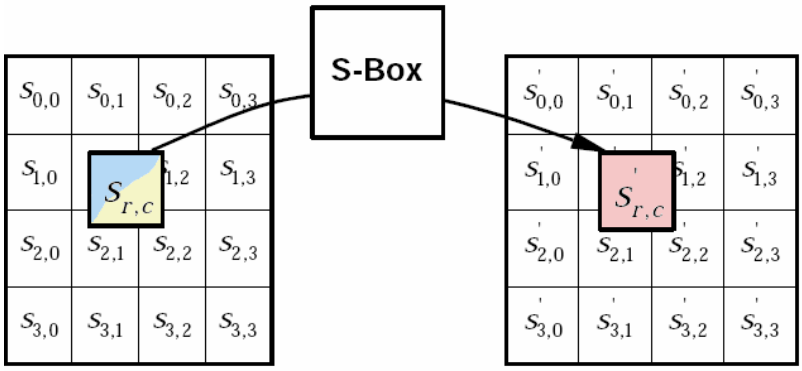
011 0000	060	48	30	0
011 0001	061	49	31	1
011 0010	062	50	32	2
011 0011	063	51	33	3
011 0100	064	52	34	4
011 0101	065	53	35	5
011 0110	066	54	36	6
011 0111	067	55	37	7
011 1000	070	56	38	8
011 1001	071	57	39	9
011 1010	072	58	3A	:
011 1011	073	59	3B	;
011 1100	074	60	3C	<
011 1101	075	61	3D	=
011 1110	076	62	3E	>
011 1111	077	63	3F	?

101 0000	120	80	50	P
101 0001	121	81	51	Q
101 0010	122	82	52	R
101 0011	123	83	53	S
101 0100	124	84	54	T
101 0101	125	85	55	U
101 0110	126	86	56	V
101 0111	127	87	57	W
101 1000	130	88	58	X
101 1001	131	89	59	Y
101 1010	132	90	5A	Z
101 1011	133	91	5B	[
101 1100	134	92	5C	\
101 1101	135	93	5D	]
101 1110	136	94	5E	^
101 1111	137	95	5F	_

111 0000	160	112	70	p
111 0001	161	113	71	q
111 0010	162	114	72	r
111 0011	163	115	73	s
111 0100	164	116	74	t
111 0101	165	117	75	u
111 0110	166	118	76	v
111 0111	167	119	77	w
111 1000	170	120	78	x
111 1001	171	121	79	y
111 1010	172	122	7A	z
111 1011	173	123	7B	{
111 1100	174	124	7C	|
111 1101	175	125	7D	}
111 1110	176	126	7E	~

Transformation SubBytes().

→transformation non linéaire appliquée indépendamment à chacun des octets de l'état en utilisant une table de substitution (Sbox).



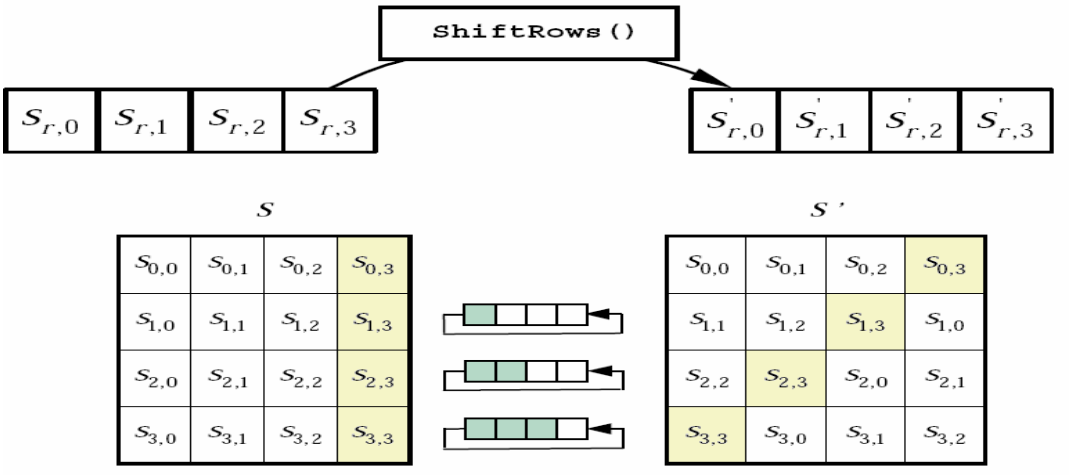
Exemple: pour $s_{1,1}$= {53}

$s'_{1,1}$ =SubBytes($s_{1,1}$)={ed}

x \ y	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

Transformation ShiftRows().

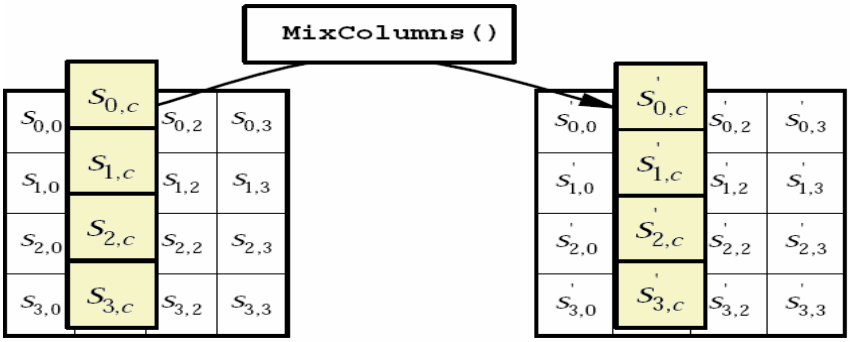
→ Permutation cyclique des octets sur les lignes de l'état. Le décalage des octets correspond à l'indice de la ligne considérée($0 \leq r < 4$).



$$s'_{r,c} = s_{r,(c+r) \bmod}$$

Transformation MixColumns().

→transformation appliquée à un état colonne après colonne.



C'est une transformation linéaire: un produit matriciel utilisant les 4 octets d'une colonne. Les colonnes sont traitées comme des polynômes dans GF(2^8) et multipliées modulox ^{4} +1 avec les polynômes fixes donnés figures suivante:

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

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$
$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$
$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$

Transformation AddRoundKey().

→ajout de la clef de ronde (ou de la clef lors de la ronde initiale) à l'état considéré (l'addition étant prise au sens ou exclusif). Un XOR (au niveau bit) est appliqué entre chacun des octets de l'état et de la clef de ronde.

XOR		
0	0	0
0	1	1
1	0	1
1	1	0

X⊕0=X

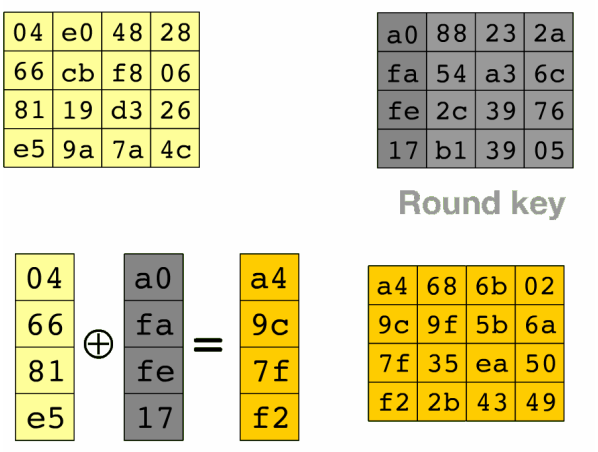
X⊕1=not(X)

X ⊕X= 0

X⊕not(X)=1

X⊕a ⊕X=a

Exemple:

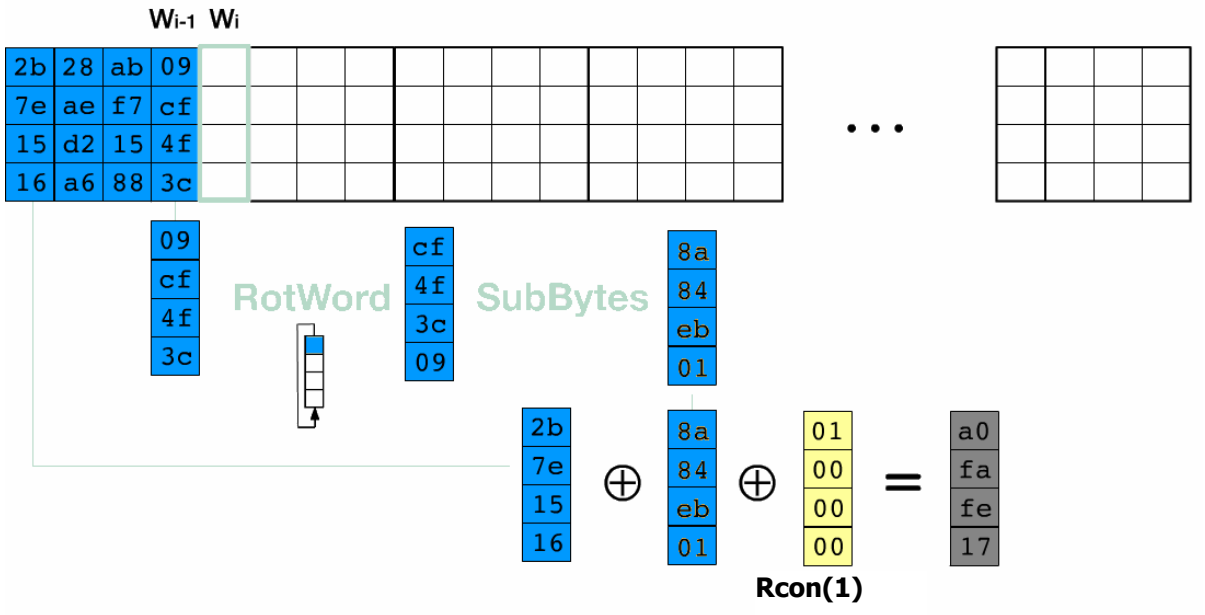


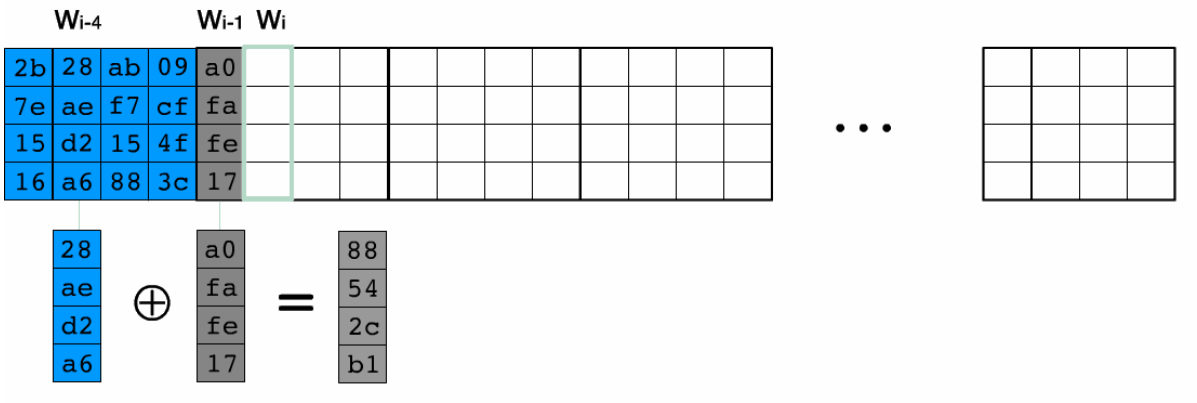
KeyExpansion–Diversification de la clef.

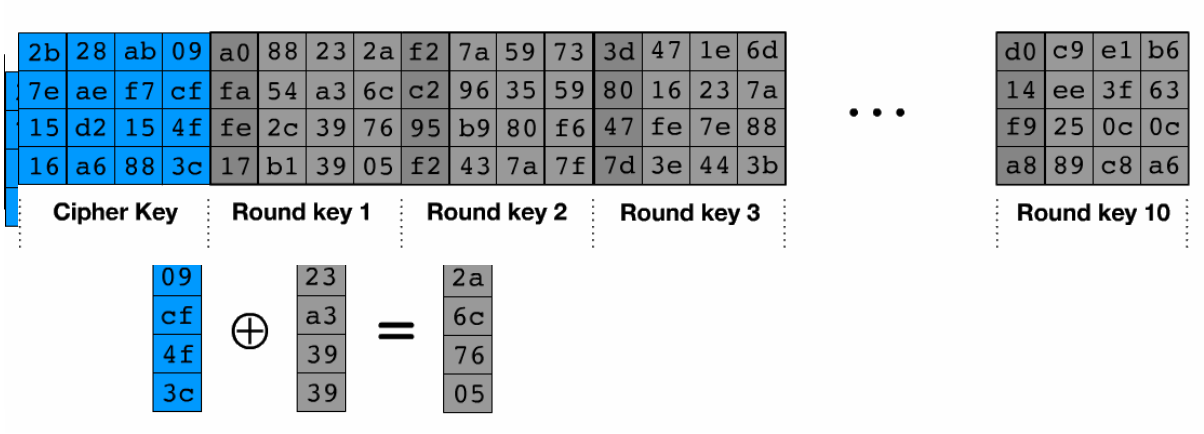
01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Rcon

2




Wi-4
Wi-1
Wi
2b 28 ab 09 a0
7e ae f7 cf fa
15 d2 15 4f fe
16 a6 88 3c 17
28 ae d2 a6
a0 fa fe 17
88 54 2c b1


2b 28 ab 09 a0 88 23 2a f2 7a 59 73 3d 47 1e 6d
7e ae f7 cf fa 54 a3 6c c2 96 35 59 80 16 23 7a
15 d2 15 4f fe 2c 39 76 95 b9 80 f6 47 fe 7e 88
16 a6 88 3c 17 b1 39 05 f2 43 7a 7f 7d 3e 44 3b
d0 c9 e1 b6
14 ee 3f 63
f9 25 0c 0c
a8 89 c8 a6
Cipher Key
Round key 1
Round key 2
Round key 3
Round key 10
09 cf 4f 3c
23 a3 39 39
2a 6c 76 05

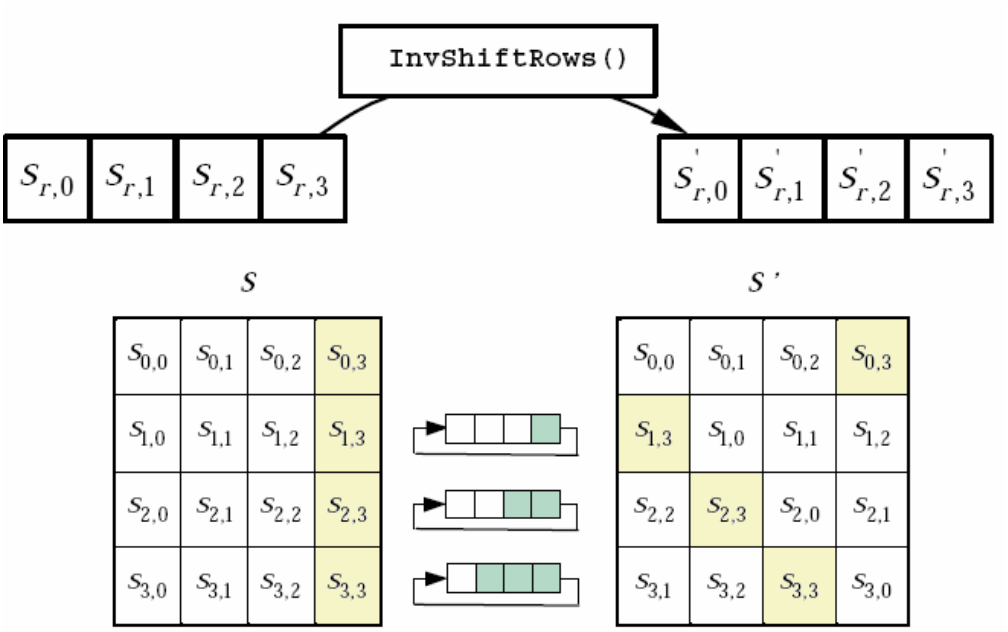
InverseCipher.

Les clefs de ronde sont utilisées dans l'ordre inverse de celui du chiffrement

.On noter a que l'ordre des transformations diffère de celui du Cipher.

→le déchiffrement consiste à appliquer dans l'ordre inverse du chiffrement les transformations inverses correspondantes (*"détricoter le chiffrement"*).

InvShiftRows().



InvSubBytes().

→inverse de la transformation SubBytes().

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

Pour $s_{i,j}=\{ed\}$

$s'_{i,j}=\text{InvSubBytes}(s_{i,j})=\{53\}$

InvMixColumns().

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

$$s'_{0,c} = (\{0e\} \bullet s_{0,c}) \oplus (\{0b\} \bullet s_{1,c}) \oplus (\{0d\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$

$$s'_{1,c} = (\{09\} \bullet s_{0,c}) \oplus (\{0e\} \bullet s_{1,c}) \oplus (\{0b\} \bullet s_{2,c}) \oplus (\{0d\} \bullet s_{3,c})$$

$$s'_{2,c} = (\{0d\} \bullet s_{0,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0e\} \bullet s_{2,c}) \oplus (\{0b\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{0b\} \bullet s_{0,c}) \oplus (\{0d\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0e\} \bullet s_{3,c})$$

Equivalent InverseCipher.

→version utilisant un ordre des transformations identique à celui du Cipher (chaque transformation étant remplacée par son inverse).

Propriétés algorithmiques:

1. Il est possible d'inverser l'ordre des transformation SubBytes() et ShiftRows(), et également pour InvSubBytes() et InvShiftRows().
2. Les transformations Mix Columns() et InvMixColumns() sont linéaires vis-à-vis de la colonne d'entrée, c'est-à-dire:
   InvMixColumns(stateXORRoundKey)=
   InvMixColumns(state)XORInvMixColumns(RoundKey)

Avantages:

- AES128 est largement pris en charge par les systèmes d'exploitation, les navigateurs web et les logiciels de chiffrement, ce qui facilite son utilisation et son intégration dans les applications.
- L'algorithme AES128 est relativement simple à mettre en œuvre, ce qui peut rendre sa configuration plus facile pour les développeurs et les administrateurs système.
- AES128 a été approuvé par le gouvernement américain pour une utilisation dans des applications de sécurité nationale, ce qui peut renforcer la confiance dans sa sécurité.

Inconvénients:

- Bien que la taille de la clé de 128 bits soit considérée comme suffisante pour la plupart des utilisations, certains experts en sécurité préfèrent des tailles de clé plus grandes pour une sécurité accrue.
- AES128 ne fournit pas d'authentification de message intégrée, ce qui signifie que des attaquants peuvent potentiellement modifier les données sans être détectés.
- Si une personne malveillante obtient accès à la clé de chiffrement AES128, elle peut déchiffrer facilement toutes les données protégées avec cette clé, ce qui souligne l'importance de la gestion de clé sécurisée.