LE CHIFFREMENT AES

1. Le contexte d'AES

En 1977, un chiffrement appelé DES fut créé par IBM. Il scindait les messages en blocs de 64 bits, pour les chiffrer avec une clé de 64 bits également. On s'aperçut qu'il comportait quelques faiblesses et il fut triplé, c'est-à-dire que les opérations de chiffrement furent appliquées 3 fois de suite : ce fut le triple DES. Puis il fut remplacé par un nouvel algorithme : l'AES.

L'AES fut adopté par le National Institute of Standards and Technology en 2000.

2. Principe de fonctionnement de l'AES

AES est un algorithme de chiffrement symétrique : il utilise la même clé pour le chiffrement et le déchiffrement des données. Il repose sur une combinaison d'opérations de substitutions et d'opérations de permutations.

Les données du message à chiffrer sont réparties en **blocs,** qui sont des carrés de 16 cases (4x4) dans lesquels ces données sont placées. On applique ensuite sur ces blocs un certain nombre d'opérations de chiffrement appelées **tours.**

Chaque tour suit une série d'étapes bien définies.

3. La structure de l'algorithme AES

Voici les étapes d'un chiffrement AES (l'exemple concret expliquera en détail les opérations) :

- Avant le premier tour, **un premier chiffrement** est effectué avec une **clé initiale**. Puis viennent les tours qui comportent les opérations suivantes :

- **SubBytes** (Substitution des octets)

Dans cette étape, chaque octet du bloc de données chiffrées est remplacé par un autre octet selon une table de substitution appelée **S-box** (voir cette table en annexe 1)

- **ShiftRows** (Permutation des lignes)

Dans cette étape, les octets de chaque ligne du bloc de données (considéré comme une matrice de 4x4) sont déplacés de manière cyclique. La première ligne n'est pas modifiée, la seconde est décalée

d'un octet vers la gauche, la troisième de deux octets, et la quatrième de trois octets. Cela permet de brouiller les positions des données dans le bloc.

- **MixColumns** (Mélange des colonnes)

Dans cette étape, chaque colonne du bloc est mélangée avec une ligne d'un autre bloc en effectuant un produit matriciel, l'un des blocs étant constituée de chiffres et l'autre contenant les données.

- **AddRoundKey** (Ajout de la clé de tour)

Enfin, une opération XOR (voir la définition dans l'exemple ci-dessous) est effectuée entre le bloc de données et une sous-clé dérivée de la clé initiale. La sous-clé change à chaque tour, elle est obtenue à l'aide d'un processus appelé **Key Expansion**. Donc à chaque round, les blocs ont une nouvelle sous-clé spécifique.

4. Le processus de Key Expansion (Expansion de la clé)

Dans AES, la clé initiale est étendue pour générer une sous-clé unique pour chaque tour. L'algorithme AES inclut **un processus d'expansion de la clé qui utilise la clé initiale et la transforme en une série de sous-clés.** Chaque sous-clé dépend de la sous-clé du round précédent : la clé du round 2 est calculé par un algorithme à partir de la clé du round 1.

5. Processus complet de chiffrement

1. **Initialisation** : La première étape est un **AddRoundKey**, où le bloc de données d'entrée est chiffré avec la clé initiale.

2. **Tours principaux** : Ensuite, chaque tour (excepté le dernier) applique les quatre étapes mentionnées plus haut : SubBytes, ShiftRows, MixColumns, et AddRoundKey.

3. **Dernier tour** : Lors du dernier tour, l'étape **MixColumns** est omise, ne laissant que SubBytes, ShiftRows, et AddRoundKey.

6. Exemple concret de chiffrement :

6.1 Initialisation : AddRoundKey

Soit le message clair :

L	**e**	**s**		**R**	**o**	**i**	**s**		**M**	**a**	**u**	**d**	**i**	**t**	**s**

et la clé :

c	**r**	**y**	**p**	**t**	**o**	**g**	**r**	**a**	**p**	**h**	**i**	**q**	**u**	**e**	**s**

Ici, la clé est un mot pour une meilleure compréhension, mais dans la réalité c'est une clé aléatoire.

On code le message et la clé en numération hexadécimale. Si vous le souhaitez, consultez la fiche n°4 « Les codes et les dictionnaires » pour voir le codage ASCII et la numération hexadécimale.

Vous trouverez également en annexe 2 une table ASCII pour une meilleure compréhension.

Cela nous donne donc en hexadécimal :

L	**e**	**s**		**R**	**o**	**i**	**s**		**M**	**a**	**u**	**d**	**i**	**t**	**s**
4C	**65**	**73**	**20**	**52**	**6F**	**69**	**73**	**20**	**4D**	**61**	**75**	**64**	**69**	**74**	**73**

et pour la clé :

c	**r**	**y**	**p**	**t**	**o**	**g**	**r**	**a**	**p**	**h**	**i**	**q**	**u**	**e**	**s**
63	**72**	**79**	**70**	**74**	**6F**	**67**	**72**	**61**	**70**	**68**	**69**	**71**	**75**	**65**	**73**

Les données en hexadécimal sont ensuite réparties dans des carrés de 4 x 4 cases, appelés blocs. Chaque bloc comporte donc 16 cases, et dans chaque case se trouve un nombre hexadécimal de 2 chiffres qui en binaire égale un octet (8 bits). On a donc au total des blocs de 128 bits (16 x8).

On écrit les données verticalement dans la colonne 1, puis colonne 2, colonne 3 etc. , ce qui donne pour le message clair:

4C	**52**	**20**	**64**
65	**6F**	**4D**	**69**
73	**69**	**61**	**74**
20	**73**	**75**	**73**

et pour la clé :

63	**74**	**61**	**71**
72	**6F**	**70**	**75**
79	**67**	**68**	**65**
70	**72**	**69**	**73**

Message clair **Clé**

Comme dans un chiffrement de type Vigenère, on va additionner le message clair et la clé.
Ceci va s'effectuer en additionnant chaque nombre du bloc du message clair avec le nombre de la case correspondante du bloc de la clé.

Pour cela, l'ordinateur convertit les nombres hexadécimaux en nombres binaires, effectue les additions en binaire, puis reconvertit le résultat des additions en hexadécimal.

Les calculs sont effectués en « chiffrement XOR » (en anglais eXclusive OR, c'est-à-dire « ou exclusif »).

L'addition XOR , en binaire, est notée $\oplus$. Les règles de calcul sont très spécifiques et reposent sur l'addition des 0 et des 1. Elles sont les suivantes :

Addition en XOR		
A	**B**	**R = A ⊕ B**
0	0	0
0	1	1
1	0	1
1	1	0

On trouvera en annexe 3 le détail des opérations d'addition en XOR. Dans notre exemple on effectue le calcul : 01001100 ⊕ 0110001 = 00101111, et donc = 2F en hexadécimal.

On a donc 4C ⊕ 63 = 2F

4C	**52**	**20**	**64**
65	**6F**	**4D**	**69**
73	**69**	**61**	**74**
20	**73**	**75**	**73**

Clair

+

63	**74**	**61**	**71**
72	**6F**	**70**	**75**
79	**67**	**68**	**65**
70	**72**	**69**	**73**

Clé

=

2F			

Chiffré

6.2 SubBytes :

Dans cette étape, chaque octet du bloc de données est remplacé par un autre octet selon une table de substitution appelée **S-box** (voir cette table en annexe 1). Cette table de substitution est conçue pour être non-linéaire, ce qui permet de rendre le chiffrement plus résistant aux attaques.

6.3 ShiftRows (permutation des lignes) :

Dans cette étape, les octets de chaque ligne du bloc de données (considéré comme une matrice de 4x4) sont déplacés de manière cyclique. La première ligne n'est pas modifiée, la seconde est décalée d'un octet vers la gauche, la troisième de deux octets, et la quatrième de trois octets. Cela permet de brouiller les positions des données dans le bloc.

En supposant les données dans un bloc de 16 cases, le tableau d'origine est ainsi modifié :

1	**2**	**3**	**4**
5	**6**	**7**	**8**
9	**10**	**11**	**12**
13	**14**	**15**	**16**

→

1	**2**	**3**	**4**
6	**7**	**8**	**5**
11	**12**	**9**	**10**
16	**13**	**14**	**15**

6.4 Mix Columns (Mélange des colonnes)

On effectue un produit matriciel : on multiplie chaque donnée d'une colonne par une ligne d'une matrice prédéfinie par l'algorithme AES.

Voici un exemple :

i_0	i_1	i_2	i_3
i_4	i_5	i_6	i_7
i_8	i_9	i_{10}	i_{11}
i_{12}	i_{13}	i_{14}	i_{15}

x

1	**2**	**3**	**4**
6	**7**	**8**	**5**
11	**12**	**9**	**10**
16	**13**	**14**	**15**

=

		c	**d**
			e
			f
			g

Pour la 1ère ligne, on calcule : $(4xi_0) + (5xi_1) + (10xi_2) + (15xi_3)$ qui est égal à **d**

Puis pour la 2ème ligne : = $(4xi_4) + (5xi_5) + (10xi_6) + (15xi_7)$ = **e**

On continue pour la 3ème ligne : $(4xi_8) + (5xi_9) + (10xi_{10}) + (15xi_{11})$ = **f**

Et on calcule enfin la valeur g de la 4ème ligne = **g**.

On recommence le processus sur la colonne 3 :

$(3xi_0) + (8xi_1) + (9xi_2) + (14xi_3)$ qui est égal à **c**, 1ère donnée de la colonne 3.
Et ainsi de suite sur les 3 autres données de la 3éme colonne, puis les colonnes 2 et 1.

Les multiplications sont également effectuées **en numération binaire** , par l'intermédiaire de calculs sur des polynômes. Le processus mathématique est un peu complexe, on trouvera si on le souhaite le détail des calculs en annexe 3.

On obtient au final un nouveau bloc qui est résultat de ce produit matriciel.

6.5 AddRoundKey

Enfin, une nouvelle opération d'addition est effectuée entre le bloc de données et une sous-clé dérivée de la clé initiale, pour préparer les opérations sur le bloc suivant. La sous-clé change à chaque tour, provenant du processus appelé **Key Expansion** (voir paragraphe 4).

On répète toutes ces opérations 9 à 13 fois avec les sous-clés dérivées de la clé principale, et on ne fait pas de **MixColumn** sur la dernière opération pour réduire le temps de chiffrement de AES.

7. Déchiffrement

Le processus de déchiffrement est une inversion des étapes de chiffrement, avec quelques différences spécifiques :

- **Inverse SubBytes** : Utilisation d'une table de substitution inverse (Inverse S-box).
- **Inverse ShiftRows** : Les lignes sont décalées dans l'autre sens.

- **Inverse MixColumns** : Application inverse du mélange des colonnes.
- **AddRoundKey** reste inchangé, puisqu' une opération XOR est son propre inverse.

8. Conclusion : sécurité d'AES

Comme toujours dans un mode de chiffrement, il faut faire un compromis entre sécurité et temps de calcul. L'étape **MixColumn**, qui est un produit matriciel, est longue en temps de calcul.

On considère actuellement qu'avec un minimum de 10 tours et avec une clé suffisamment longue d'au moins 256 bits, AES est bien sécurisé. Aucune attaque pratique ne remet en cause la sécurité d'AES lorsqu'il est correctement utilisé. (Un nombre de 256 bits représente un nombre d'environ 77 chiffres en décimal)

AES est largement utilisé dans de nombreuses applications (VPN, https, chiffrement des fichiers, etc.). Il a été conçu pour être efficace aussi bien sur le matériel que sur le logiciel, ce qui le rend adapté pour un large éventail de systèmes, des serveurs puissants aux dispositifs embarqués.

9. Remerciements :

Pour réaliser cette fiche, j'ai utilisé, parmi d'autres sources d'information, une **vidéo réalisée par Mickaël Dupont**, publiée sur YouTube (et citée avec l'autorisation de l'auteur) :

https://www.youtube.com/@MickaDupont?app=desktop

Je vous la recommande vivement, elle est extrêmement claire et pédagogique.

*****

***

*

Annexe 1

	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

Annexe 2

Table ASCII (0 - 127)

Binaire	Hex.	Déc.	Caractères ASCII	Explications	Groupe
		0-31			Caractère de contrôle
0100000	20	32	SP	Espace (*Space*)	Caractère spécial
0100001	21	33	!	Point d’exclamation	Caractère spécial
0100010	22	34	"	Guillemets droits en haut	Caractère spécial
0100011	23	35	#	Dièse	Caractère spécial
0100100	24	36	$	Signe dollar	Caractère spécial
0100101	25	37	%	Signe pourcentage	Caractère spécial
0100110	26	38	&	Esperluette	Caractère spécial
0100111	27	39	‘	Apostrophe	Caractère spécial
0101000	28	40	(	Parenthèse gauche	Caractère spécial
0101001	29	41	)	Parenthèse droite	Caractère spécial
0101010	2A	42	*	Astérisque	Caractère spécial
0101011	2B	43	+	Signe plus	Caractère spécial
0101100	2C	44	,	Virgule	Caractère spécial
0101101	2D	45	-	Trait d’union	Caractère spécial
0101110	2E	46	.	Point (fin de phrase)	Caractère spécial
0101111	2F	47	/	Barre oblique (« slash »)	Caractère spécial
0110000	30	48	0		Chiffre
0110001	31	49	1		Chiffre
0110010	32	50	2		Chiffre
0110011	33	51	3		Chiffre
0110100	34	52	4		Chiffre
0110101	35	53	5		Chiffre
0110110	36	54	6		Chiffre
0110111	37	55	7		Chiffre
0111000	38	56	8		Chiffre
0111001	39	57	9		Chiffre
0111010	3A	58	:	Deux points	Caractère spécial

0111011	3B	59	;	Point-virgule	Caractère spécial
0111100	3C	60	<	Inférieur à	Caractère spécial
0111101	3D	61	=	Signe égal	Caractère spécial
0111110	3E	62	>	Plus grand que	Caractère spécial
0111111	3F	63	?	Point d’interrogation	Caractère spécial
1000000	40	64	@	Arobase	Caractère spécial
1000001	41	65	A		Lettre majuscule
1000010	42	66	B		Lettre majuscule
1000011	43	67	C		Lettre majuscule
1000100	44	68	D		Lettre majuscule
1000101	45	69	E		Lettre majuscule
1000110	46	70	F		Lettre majuscule
1000111	47	71	G		Lettre majuscule
1001000	48	72	H		Lettre majuscule
1001001	49	73	I		Lettre majuscule
1001010	4A	74	J		Lettre majuscule
1001011	4B	75	K		Lettre majuscule
1001100	4C	76	L		Lettre majuscule
1001101	4D	77	M		Lettre majuscule
1001110	4E	78	N		Lettre majuscule
1001111	4F	79	O		Lettre majuscule
1010000	50	80	P		Lettre majuscule
1010001	51	81	Q		Lettre majuscule
1010010	52	82	R		Lettre majuscule
1010011	53	83	S		Lettre majuscule
1010100	54	84	T		Lettre majuscule
1010101	55	85	U		Lettre majuscule
1010110	56	86	V		Lettre majuscule
1010111	57	87	W		Lettre majuscule
1011000	58	88	X		Lettre majuscule
1011001	59	89	Y		Lettre majuscule
1011010	5A	90	Z		Lettre majuscule
1011011	5B	91	[	Crochet gauche	Caractère spécial
1011100	5C	92	\	Barre oblique inversée (*backslash*)	Caractère spécial
1011101	5D	93	]	Crochet droit	Caractère spécial
1011110	5E	94	^	Accent circonflexe	Caractère spécial
1011111	5F	95	_	Tiret bas	Caractère spécial
1100000	60	96	`	Accent grave	Caractère spécial
1100001	61	97	a		Lettre minuscule
1100010	62	98	b		Lettre minuscule
1100011	63	99	c		Lettre minuscule
1100100	64	100	d		Lettre minuscule
1100101	65	101	e		Lettre minuscule
1100110	66	102	f		Lettre minuscule

1100111	67	103	g		Lettre minuscule
1101000	68	104	h		Lettre minuscule
1101001	69	105	i		Lettre minuscule
1101010	6A	106	j		Lettre minuscule
1101011	6B	107	k		Lettre minuscule
1101100	6C	108	l		Lettre minuscule
1101101	6D	109	m		Lettre minuscule
1101110	6E	110	n		Lettre minuscule
1101111	6F	111	o		Lettre minuscule
1110000	70	112	p		Lettre minuscule
1110001	71	113	q		Lettre minuscule
1110010	72	114	r		Lettre minuscule
1110011	73	115	s		Lettre minuscule
1110100	74	116	t		Lettre minuscule
1110101	75	117	u		Lettre minuscule
1110110	76	118	v		Lettre minuscule
1110111	77	119	w		Lettre minuscule
1111000	78	120	x		Lettre minuscule
1111001	79	121	y		Lettre minuscule
1111010	7A	122	z		Lettre minuscule
1111011	7B	123	{	Accolade gauche	Caractère spécial
1111100	7C	124	|	Trait vertical (*pipe*)	Caractère spécial
1111101	7D	125	}	Accolade droite	Caractère spécial
1111110	7E	126	~	Tilde	Caractère spécial
111111	7F	127	DEL	Delete	Caractère spécial

Annexe 3

Les opérations en numération binaire et hexadécimale

1. Les systèmes de numération décimale, binaire et hexadécimale :

Un ordinateur calcule beaucoup plus vite en système binaire qu'en système décimal. C'est pourquoi en informatique on effectue les calculs en système binaire.

La relation entre les 3 systèmes de numération est la suivante :

Numération décimale : le nombre 523, par exemple, est constitué ainsi :

Position	10^2	10^1	10^0
Nombre	5	2	3

On a donc $523 = (5x10^2) + (2x10^1) + (3x10^0)$

$= (5x100) + (2x10) + (3x1)$

$= 500 + 20 + 3$

$= 523$

Numération hexadécimale :

Soit le nombre hexadécimal 5C. Comme pour le décimal, on a :

Position	16^1	16^0
Nombre	5	C

De la même façon que pour le décimal, on a :

$5C = (5x16^1) + (Cx16^0)$

$= (5x16) + (12x1)$ en décimal

$= 80 + 12 = 92$

Numération binaire

Soit le nombre 203 qui s'écrit en binaire binaire 11001011 :

Position du bit	7	6	5	4	3	2	1	0
Nombre	1	1	0	0	1	0	1	1

En décimal, il se calcule comme suit :

$= 2^7 + 2^6 + 2^3 + 2^1 + 2^0$

$= 128 + 64 + 8 + 2 + 1 = 203$

Relation entre le binaire et l'hexadécimal :

Si l'on considère un octet, c'est à dire 8 bits, on peut constater qu'un nombre binaire de 8 bits est constitué de 4 bits de rang 7 à 4 et de 4 bits de rang 3 à 0. Le nombre décimal 203 va donc s'écrire

Rang du bit	**7 6 5 4**	**3 2 1 0**
Binaire	1 1 0 0	1 0 1 1
Hexadécimal	C	B

On a séparé en deux groupes de 4 pour plus de clarté, mais bien sûr c'est un seul et même chiffre d'un octet. Les 4 derniers bits sont 1011 qui est égal à B et les 4 premiers bits sont 1100 qui est égal à C. On a donc CB = en décimal (12 x16) +11 = 192 + 11 = 203 que l'on retrouve bien.

Autre exemple :

Rang du bit	**7 6 5 4**	**3 2 1 0**
Binaire	0 1 1 0	1 0 0 1
Hexadécimal	6	9

Et 69 en hexadécimal = (6 x 16) + 9 =105 en décimal

Dans le code ASCII , on peut ainsi avoir 256 caractères (de 0 à 255) représentés chacun par un octet en binaire ou un nombre de 2 chiffres en hexadécimal.

On a ainsi de 0 à 255 :

	Décimal	**Binaire**	**Hexadécimal**
de	0	00000000	00
	15	00001111	0F
	16	00010000	10
	127	01111111	7F
	128	10000000	80
à	255	11111111	FF

2. L'addition XOR en binaire :

L'addition XOR , en binaire, est notée ⊕. En XOR, les règles de calcul sont les suivantes:

Addition en XOR		
A	**B**	**R = A ⊕ B**
0	0	0
0	1	1
1	0	1
1	1	0

Notons qu' il n'y a pas de retenues, contrairement à une addition classique de 2 nombres binaires.

Ce qui nous donne par exemple :

	Hexa		**Binaire**
	4C	--->	01001100
⊕	63	--->	⊕ 01100011
	-----		----------
	2F	←---	**00101111**

3. La multiplication en binaire

Dans l'opération **Mix Columns ,** il faut effectuer un produit entre les deux tableaux. Mathématiquement, c'est un produit de matrices. Voici le détail du calcul :

On effectue ce que l'on appelle un produit matriciel : on multiplie chaque donnée d'une colonne par une ligne d'une matrice prédéfinie par l'algorithme AES.

Exemple : Soit à multiplier les deux tableaux :

03	02	01	01
01	02	03	01
03	01	02	01
02	01	01	03

Matrice prédéfinie

x

FA		**31**	**C7**
C5	**2F**	**3F**	**36**
31	**12**	**9**	**10**
6D	**13**	**14**	**15**

Message

=

d		
e		
f		
g		

Résultat crypté

Pour effectuer ce produit matriciel, il faut calculer dans la case d:

d = (03xFA) + (02xC5) + ((01x31) + 01x6D)

Calcul du premier facteur

On travaille en binaire. On a : 03 x FA = 11 x 11111010
hexa binaire

Pour effectuer ce calcul, on introduit un polynôme f(x) dont les exposants de x vont être déterminés par la position des 1 dans le nombre binaire. Les exposants de x varient de 7 à 1 :

--→ 11 correspond à un polynôme : $(x^1 + x^0)$ c'est à dire (x + 1)

— > 11111010 correspond à un polynôme : $(x^7 + x^6 + x^5 + x^4 + x^3 + x)$

On voit que les exposants de x correspondent aux position des 1 dans l'octet. Il y a dans le chiffre binaire 11111010 des zéros en position 3 et en position 1, c'est à dire $x^2 + x^0$, donc ces 2 termes sont supprimés dans le polynôme.

On va donc effectuer la multiplication :

$$(x + 1)(x^7 + x^6 + x^5 + x^4 + x^3 + x)$$

Ce qui nous donne :

$x(x^7 + x^6 + x^5 + x^4 + x^3 + x) + (x^7 + x^6 + x^5 + x^4 + x^3 + x)$

$= x^8 + x^7 + x^6 + x^5 + x^4 + x^2 + x^7 + x^6 + x^5 + x^4 + x^3 + x$

Là, on supprime les valeurs de x qui sont en double, c'est à dire qui ont le même exposant. En effet, l'addition est un XOR, et donc l'addition de deux nombres identique = 0. (Règle 1 + 1 = 0). Il reste

03 x FA = $x^8 + x^3 + x^2 + x$

Afin de supprimer les termes en x^8, on soustrait le polynôme irréductible $(x^8 + x^4 + x^3 + x + 1)$

Ce qui nous donne :

03 x FA = $x^8 + x^3 + x^2 + x - (x^8 + x^4 + x^3 + x + 1)$

03 x FA = $x^4 + x^2 + 1$

Soit en résultat binaire 03 x FA = 00010101

On effectue le même type de calcul pour 02xC, soit en binaire = 10 x 11000101

Pour la 2ème multiplication, le même type de calcul avec les polynômes, que nous ne détaillerons pas de nouveau, nous donne le résultat :

02 x C5 = $x^7 + x^4 + 1$ soit en binaire = 10010001

Pour les 3ème et 4ème termes, c'est plus facile puisque 01 x 31 = 31 et 01 x 6D = 6D.

01 x 31 = 00110001
01 x 6D = 01101101

En définitive, le calcul de d est le suivant (en XOR)

```
    03 x FA  = 00010101
⊕   02 x C5  = 10010001
⊕   01 x 31  = 00110001
⊕   01 x 6D  = 01101101
               ----------
         d =   11011000
```

Soit d = D8 en hexadécimal, c'est ce nombre qui est reporté dans le tableau du résultat chiffré.

L'algorithme effectue le même calcul de multiplication de matrices sur chacune des 16 cases du tableau.

*****
***
*