LA CRYPTOLOGIE MODERNE

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**

Message clair

et pour la clé :

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

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 $\oplus$ 0110001 = 00101111, et donc = 2F en hexadécimal.

On a donc 4C $\oplus$ 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 :

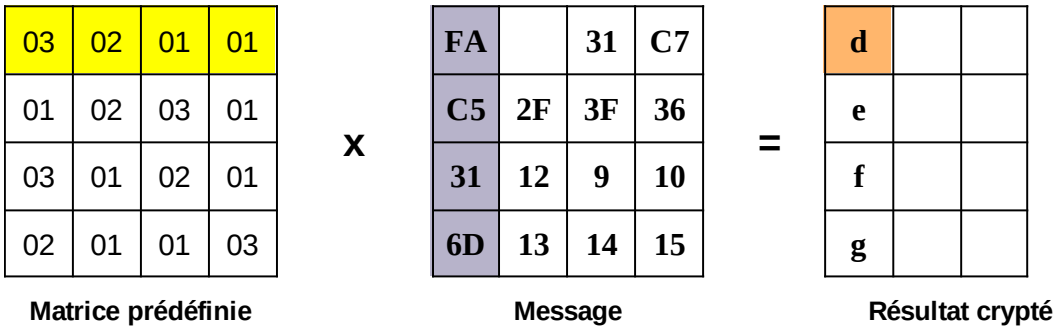
	Hexa		**Binaire**
	4C	--->	01001100
⊕	63	--->	⊕ 01100011
	-----		----------
	2F	←---	**00101111**

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 :



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 0 :

--→ 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 :

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

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

Soit en résultat binaire 03 x FA = 00010101

On effectue le même type de calcul pour 02xC5, 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 :

$02 \times C5 = x^7 + x^4 + 1$ 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.

*****
***
*

LE CHIFFREMENT PAR COURBES ELLIPTIQUES

1. **Qu'est ce qu'une courbe elliptique ?**

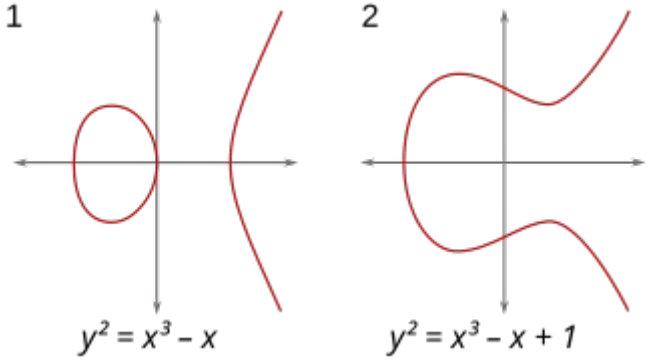
Les courbes elliptiques sont des fonctions de type : $y^2 = f(x)$ où f(x) est un polynôme en x^3

En cryptographie, on utilise des courbes elliptiques de la forme :

$$y^2 = x^3 + ax + b$$

Remarquons qu'il n'y a pas de x^2 . Les coefficients a et b sont des nombres réels. Selon le choix de ces coefficients, les graphes peuvent avoir deux formes possibles.

Exemples :



On voit que dans le graphe 1, l'équation a trois racines réelles distinctes (-1, 0, et +1) et que dans le graphe 2, elle n'a qu'une seule racine réelle.

2. **Utilisation des courbes elliptiques en cryptographie :**

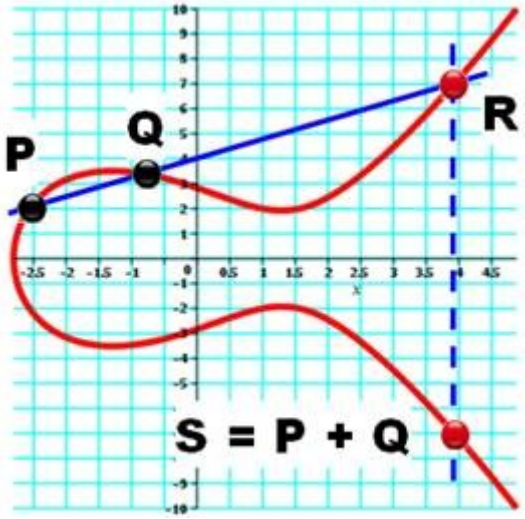
Cette fiche a pour but de présenter de façon simple le chiffrement par courbes elliptiques. Nous resterons dans un contexte général et ne tiendrons pas compte des nombreux cas particuliers présentés par ces courbes. Nous essaierons de rester au niveau de mathématiques du lycée.

2.1 Addition de deux points sur une courbe elliptique :

Pour commencer, on définit l'addition de deux points de la courbe. L'addition s'effectue de la manière suivante : on choisit deux points P et Q sur une courbe elliptique. Ces deux points sont définis par leurs coordonnées $P(x_1, y_1)$ et $Q(x_2,y_2)$.

Deux points P et Q sur une courbe elliptique forment une droite qui coupe toujours la courbe en un troisième point. Soit R ce point. Le symétrique du point R par rapport à l'axe des x, appelons-le S, **est défini comme la somme des points P et Q**. Les points sont définis par leurs coordonnées.

Cette définition peut paraître un peu étrange, regardons un graphe :



On a donc $S(x_3, y_3) = P(x_1, y_1) + Q(x_2,y_2)$

Pour ne pas alourdir cette présentation, nous verrons en détail l'aspect mathématique de cette addition dans l'annexe 1. Pour l'instant prenons cette addition telle qu'elle est.

Sur une courbe elliptique, de même que l'on peut additionner deux points, on peut également multiplier un point par un nombre entier.

2.2 Multiplication d'un point par un entier :

Puisque l'on a défini l'addition, continuons et définissons la multiplication d'un point par un nombre entier d.

Si P est un point sur la courbe, on peut calculer 2P, 3P, 4P etc. en utilisant ce qu'on appelle la *multiplication scalaire*. Concrètement **on répète le principe de l'addition** : P + P = 2P, puis 2P + P = 3P, puis 3P + P = 4P et ainsi de suite…

Le résultat de cette multiplication est un point S. Là encore nous verrons tout cela en détail en annexe avec des exemples chiffrés et des graphes. Le but ici est de comprendre les principes.
En résumé, nous avons donc un point **P** sur une courbe elliptique, un nombre entier **d** et un point **S** sur cette courbe tel que :

$$\mathbf{S = d \times P}$$

Et nous en arrivons enfin au principe de chiffrement :

3. Principe de chiffrement :

Le chiffrement repose sur le fait que si l'on connaît un point P sur une courbe elliptique et un autre point S tel que S = dP, il est extrêmement difficile de trouver d à partir des coordonnées de P et de S.

Ce chiffrement, qui est **asymétrique**, s'effectue donc comme suit :

- Alice et Bob se mettent d'accord, publiquement, sur une courbe elliptique ainsi que sur un point P (x_1, y_1) situé sur la courbe, ce point P étant également connu publiquement.

- Alice choisit secrètement un nombre entier d_A et envoie à Bob les coordonnées du point d_AP.

- Bob choisit secrètement un nombre entier d_B et envoie à Alice les coordonnées du point d_BP.

- Alice peut calculer $d_A(d_BP)$ et Bob peut calculer $d_B(d_AP)$, **c'est à dire $((d_Ad_B)P)$ qui est leur clé commune.**

Si Ève, qui espionne Alice et Bob, a intercepté les échanges, elle connaît l'équation de la courbe, le point P, d_AP et d_BP. Mais elle ne peut pas calculer d_A et d_B, et donc le produit $(d_Ad_B)P$ qui est la clé commune.

Le calcul de d_A en connaissant P et le produit d_AP s'appelle résoudre le *logarithme discret* sur une courbe elliptique. Si les nombres d_A et d_B sont suffisamment grands, on ne peut pas les calculer avec les performances actuelles des ordinateurs.

Ce mode de chiffrement asymétrique, comme le RSA, repose sur une difficulté arithmétique non calculable actuellement par les ordinateurs.

ANNEXE 1

NOMBRES RÉELS ET CORPS FINIS

1. Courbes elliptiques sur les nombres réels (définition mathématique générale)

Les courbes elliptiques ont d'abord été étudiées dans le cadre des mathématiques pures, sur des corps tels que les **nombres réels R**. Ce sont des objets géométriques qui possèdent une structure algébrique particulière, souvent définis par une équation de la forme :

$$y^2 = x^3 + ax + b$$

Dans ce cadre, les courbes elliptiques sont représentées comme des objets continus, avec une courbe lisse qui peut être visualisée graphiquement. Travailler sur les nombres réels ou complexes permet d'étudier les propriétés géométriques et algébriques de ces courbes (points d'inflexion, tangentes, symétries, etc.).

Cependant, en matière de cryptographie, **travailler avec des nombres réels n'est pas sécurisé ni pratique** pour plusieurs raisons :

- Les nombres réels impliquent une précision infinie, ce qui est impossible à gérer en informatique.
- Les algorithmes de chiffrement nécessitent des opérations discrètes pour éviter les attaques basées sur des approximations ou des erreurs de calcul.

Ainsi, bien que la définition mathématique des courbes elliptiques soit souvent donnée dans le contexte des nombres réels, **ce n'est pas un cadre approprié pour les applications cryptographiques**.

2. Courbes elliptiques sur un corps fini F_p

En cryptographie, pour rendre les courbes elliptiques utilisables et sécurisées, on les utilise sur un **corps fini,** souvent noté $\mathbf{F_p}$ (F pour Field = champ fini en anglais) et où p est un nombre premier. Travailler sur un corps fini signifie que toutes les opérations (addition, multiplication, etc.) se font avec des entiers **modulo p**.

Par exemple, l'équation de la courbe elliptique devient :

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

Ce passage des nombres réels à un corps fini présente plusieurs avantages :

1. **Sécurité** : Les courbes elliptiques sur les corps finis offrent une grande difficulté à résoudre certains problèmes mathématiques sous-jacents (comme le problème du logarithme discret elliptique), ce qui les rend particulièrement adaptées pour la cryptographie. Les solutions aux problèmes cryptographiques dans un corps fini sont beaucoup plus difficiles à calculer que dans R surtout avec des nombres très grands.
2. **Calculs discrets** : Dans un corps fini, tous les calculs sont discrets et bornés. Cela permet d'éviter les problèmes d'approximation liés aux nombres réels et d'utiliser des algorithmes efficaces pour les opérations de chiffrement et de déchiffrement.
3. **Efficacité** : Les opérations sur les corps finis peuvent être implémentées efficacement sur des ordinateurs, car elles se réduisent à des opérations arithmétiques simples (addition, multiplication) modulo un nombre premier p. Cela est particulièrement important dans les systèmes cryptographiques modernes, où la vitesse et la précision des calculs sont critiques.

3. Exemple concret d'utilisation d'un corps fini

Si on utilise la courbe elliptique définie par l'équation :

$$y^2 = x^3 + ax + b$$

et que l'on choisit de travailler sur le corps fini F_p, cela signifie que l'on va "restreindre" les valeurs possibles de x et y aux nombres entiers compris entre 0 et $p-1$, en appliquant des opérations modulo p. Par exemple, si p=17, alors x et y seront des entiers entre 0 et 16, et toutes les additions, multiplications et divisions seront effectuées modulo 17. Cet exemple sera traité en détail dans l'annexe 2.

Pourquoi cette distinction est-elle importante ?

- **Contexte mathématique général** : Les courbes elliptiques définies sur les nombres réels servent surtout à étudier les propriétés générales des courbes et à comprendre leur structure géométrique. C'est un contexte important pour la recherche théorique en mathématiques.

- **Contexte cryptographique** : En cryptographie, on travaille sur des **corps finis** pour garantir que les opérations soient discrètes, efficaces et sécurisées. Les courbes sur les corps finis sont celles qui résistent aux attaques cryptographiques.

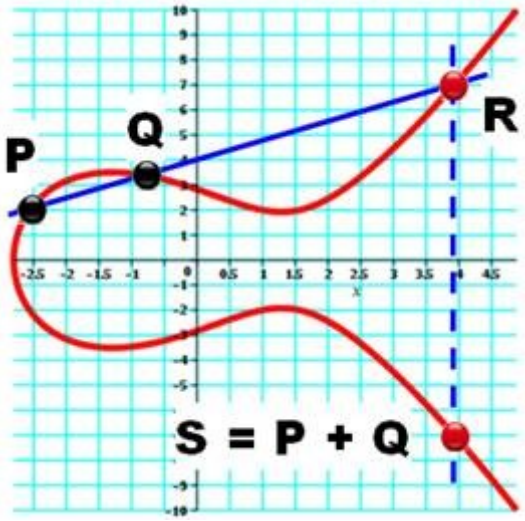
ANNEXE 2

LES OPÉRATIONS D'ADDITION ET DE MULTIPLICATION DE POINTS SUR UNE COURBE ELLIPTIQUE

Nous allons d'abord travailler sur les nombres réels. Nous avons vu que sur une courbe elliptique, on peut additionner deux points ou multiplier un point par un nombre entier. Commençons par l'addition de deux points :

1. Addition de deux points :

Voici un graphe pour visualiser :



Pour additionner deux points P + Q, on considère le point R qui est le point d'intersection de la droite passant par P et Q avec la courbe elliptique. Puis on prend le symétrique de R par rapport à l'axe des x, ce qui détermine le point S. On démontre que S = P +Q.

Nous allons expliquer dans ce paragraphe **l'aspect mathématique théorique** (voir annexe 1)**.** Puis, dans le paragraphe 2, qui présente la multiplication d'un point par un nombre, nous prendrons un exemple chiffré. Nous avons donc comme données :

- une courbe elliptique $y^2 = x^3 + ax + b$

- une droite $y = kx + m$ qui coupe la courbe en 3 points, P, Q et R

Les nombres a, b, k et m sont connus. On a défini les coordonnées de P et Q, et nous cherchons celles du point R et du point S.

On pose $P = (x_1, y_1)$, $Q = (x_2, y_2)$, $R = (x_3, y_3)$ et $S = (x_4, y_4)$

et la pente de la droite est $k = \frac{y2 - y1}{x2 - x1}$

Le point R (x_3, y_3) étant à une intersection de la courbe et de la droite, on commence par déterminer la valeur de x_3 à partir des coordonnées connues des points P et Q qui se trouvent sur la courbe et sur la droite. Après un calcul assez long, on obtient :

$$\mathbf{x_3 = k^2 - x_1 - x_2}$$

Nous avons donc déterminé la valeur de x_3 à partir des données connues k, x_1 et x_2

Puis on calcule la valeur de y_3 :

La pente k de la droite peut s'écrire également $k = \frac{y3 - y1}{x3 - x1}$

On a donc $k (x_3 - x_1) = y_3 - y_1$

Soit $\mathbf{y_3 = k (x_3 - x_1) + y_1}$

Pour terminer, nous obtenons donc les coordonnées du point S = P + Q, sachant que par définition le point S (x_4, y_4) est le symétrique de R par rapport à l'axe des abscisses.

Nous avons donc $\mathbf{x_4 = x_3}$ et $\mathbf{y_4 = - y_3 = k (x_1 - x_3) - y_1}$

En conclusion, on peut écrire le résultat de l'addition :

$$\mathbf{P (x_1,y_1) + Q (x_2,y_2) = S(x_4,y_4)}$$

Nous verrons dans le paragraphe suivant un exemple concret avec des chiffres.

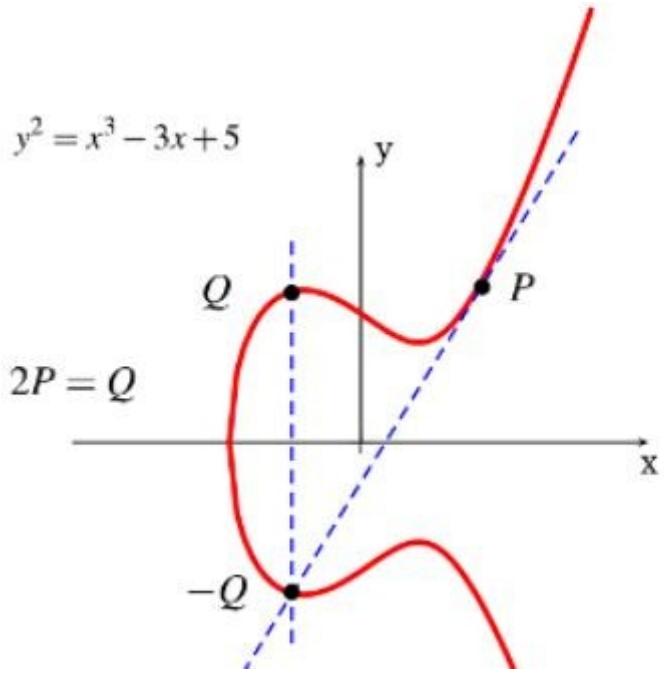
2. Multiplication d'un point P par un nombre entier

Nous allons maintenant voir **un exemple sur un corps fini** (voir annexe 1) avec des petits nombres.

Nous allons donc multiplier le point P par un nombre. Cette multiplication s'effectue selon le principe de l'addition : on additionne P + P = 2.P, puis 2P + P = 3.P, puis 3 P + P = 4.P etc.

Dans l'addition, on additionne deux points P et Q. Mais ici, on n'a qu'un seul point P. Comment faire ?

Pour ajouter P à lui-même , on trace la tangente à la courbe en P, et on considère que cette tangente touche deux fois la courbe au point P. On calcule les coordonnées de la troisième intersection, puis celles du point symétrique, comme dans l'addition. On obtient ainsi P + P = 2P.
Graphe :



Sur ce graphe, on a tracé la tangente en P à la courbe. Elle recoupe la courbe en un point – Q. Le symétrique de ce point – Q par rapport à l'axe des abscisses est le point Q. On a donc Q = 2.P.

L'équation de la courbe dans l'exemple chiffré n'est pas celle du graphe. Le graphe est simplement là pour visualiser les calculs.

Exemple chiffré en travaillant sur un corps fini:

Les données sont :

- une courbe elliptique $\mathbf{y^2 = x^3 + 2x + 2 \bmod 17}$, donc **a = 2 et b = 2**

- un point générateur **P = (5,1)**

- on effectue les calculs modulo p, avec p nombre premier, et on choisit **p = 17**

2.1 Doublement du point P :

On commence par doubler P (5,1). Comme dans le cas de l'addition, il faut déterminer le pente de la droite. La pente k de la tangente en P est donnée par :

$$k = \frac{3x2+a}{2y1} \quad \text{modulo p}$$

Il faut lire $(3 x_1^2 + a) / (2y_1)$, où x_1 et y_1 sont les coordonnées du point P (5,1)

On effectue le calcul : $k = (3*5^2 + 2) / (2*1) \bmod 17 = \frac{77}{2} \bmod 17$

Ici il faut calculer l'inverse modulaire de 2 mod 17, et on trouve 9.

Si l'on est peu familier avec la notion d'inverse modulaire, voir la fiche sur le chiffrement RSA.
Par ailleurs, le site dcode.fr permet de calculer l'inverse modulaire rapidement sur des grands nombres : https://www.dcode.fr/inverse-modulaire

On a donc $k = 77*9 \bmod 17 = 693 \bmod 17 = 13$

La pente de la tangente est **k = 13 mod 17**

On calcule ensuite les coordonnées du nouveau point avec les mêmes formules que l'addition :

$x_2 = k^2 - 2x_1 \bmod 17$, soit

$x_2 = 13^2 - 2*5 \bmod 17 = (169 - 10) \bmod 17 = 159 \bmod 17 = 6 \bmod 17$

x_2 = 6 mod 17

Puis on calcule y_2 toujours de la même façon que dans l'addition :

$y_2 = k(x_1 - x_2) - y_1 \pmod{17}$, soit

$y_2 = 13*(5-6) -1 \pmod{17}$

$= -14 \pmod{17}$

y_2 = 3 mod p soit en définitive **Q = 2.P = (6, 3)**

2. 2 Suite de la multiplication : calcul de 3.P

On veut continuer le processus de multiplication, c'est-à-dire calculer 3P. Pour ce faire, on applique les règles de l'addition de deux points, c'est à dire dire que P + 2P = 3P. On va donc définir un point S tel que S = P + Q = P + 2P = 3P

On effectue donc l'addition des 2 points P (5,1) et Q (6,3)

La pente de la droite P, Q est donnée par

$$k = \frac{1-3}{5-6} \mod 17, \text{ soit}$$

Or (1-3) = -2 et (5 -6) = -1 , soit $\frac{-2}{-1} = 2 \mod 17$

$k = 2 \mod 17$

À partir de k, on calcule $x_3 = k^2 - x_1 - x_2 \mod 17$

$x_3 = (2^2 - 5 - 6) \mod 17 = (-7) \mod 17 =$ **10 mod 17**

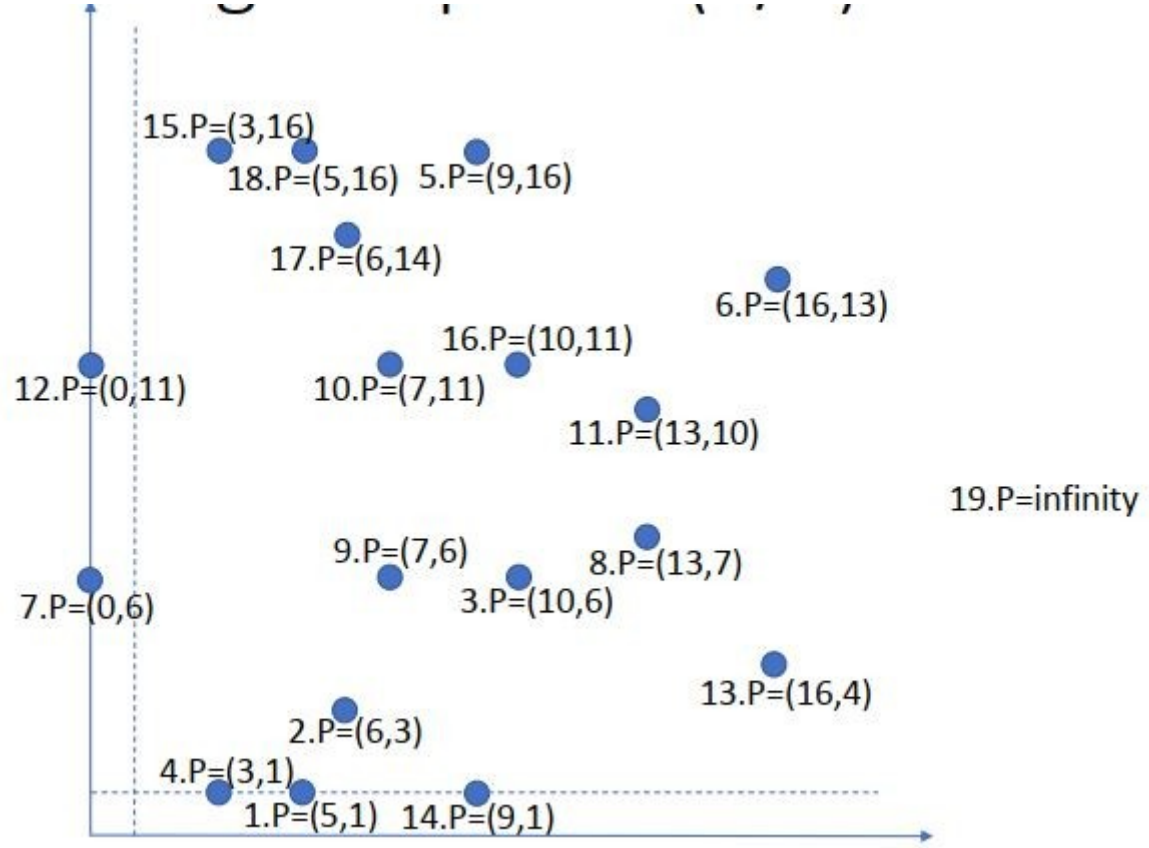
Puis on calcule $y_3 = k(x_1 - x_3) - y_1 \pmod{17}$

$y_3 = 2(5-10) - 1 \mod 10$ soit $(-11) \mod 17 =$ **6 mod 17**

Au final on a S = 3.P = (10,6)

On a vu dans le corps principal de la fiche que d'une façon générale, on avait S = dP. Dans la réalité, le chiffre d est très grand et la difficulté pour casser ce chiffre réside donc dans la difficulté de calculer d en connaissant P et S. De plus, il y a deux nombres d, celui d'Alice et celui de Bob.

Le tableau suivant permet de visualiser la position des points 4.P, 5.P, 6.P , etc. jusqu'à 18.P à partir de P (5,1), 2.P (6,3) et 3.P (10,6) :



Ce tableau est tiré du cours du professeur Alexandre Guitton, qui est téléchargeable librement sur Internet :

https://perso.isima.fr/~alguitto/index.html (voir support de cours, courbes elliptiques V2).

Le point 19.P = l'infini correspond à une droite verticale, parallèle à l'axe des y et qui coupe donc la courbe en 2 points. Le 3ème point est renvoyé à l'infini. Ce cas particulier n' a pas été traité pour ne pas alourdir la fiche.

On pourra également si on le souhaite effectuer des calculs sur le site :

https://andrea.corbellini.name/ecc/interactive/modk-add.html

Ce site permet d'effectuer des additions et des multiplications de points sur l'ensemble R des nombres réels ou sur un champ fini F_p en choisissant la courbe elliptique, le nombre premier p et les coordonnées du point P. Bien qu'en anglais, son utilisation est très simple et les graphiques qui s'affichent instantanément permettent de bien comprendre et de voir la différence entre un calcul avec les nombres réels et un calcul avec un corps fini.

3. Conclusion :

Dans les chiffrements par courbes elliptiques, il faut bien voir que dans la réalité, la clé privée d_A d'Alice et la clé privée d_B de Bob sont des nombres entiers qui ont généralement une taille d'environ 256 bits.

Comme on l'a vu au début de la fiche et dans cette annexe, le principe de chiffrement repose sur le fait qu'après l'échange de leurs clés d_A et d_B, Alice et Bob ont pu calculer une clé commune :

$$K = d_A * d_B * P$$

K est donc un point sur la courbe elliptique représenté par ses coordonnées x_K et y_K. Chacune de ces coordonnées est un entier de 256 bits. Le total des deux coordonnées représente donc 512 bits.

En pratique, souvent, seules les coordonnées x_K ou des dérivés de x_K sont utilisées afin de produire une clé symétrique plus petite. Par exemple, Alice et Bob peuvent utiliser une clé de chiffrement AES de 256 bits en utilisant x_K comme clé initiale pour dériver des clés, puisque AES utilise des clés dérivées à partir de la clé initiale (voir fiche sur le chiffrement AES).

Pour se faire une idée de la taille de ces clés, rappelons qu'un nombre de 256 bits en binaire représente un nombre entier d'environ 77 chiffres en décimal et de 64 chiffres en hexadécimal.

*****
***
*

LE CHIFFREMENT HYBRIDE :

1. Le chiffrement hybride :

Le chiffrement hybride consiste à associer deux types de chiffrement : un chiffrement symétrique (type AES) et un chiffrement asymétrique (type RSA). Rappelons la nature de ces deux types de chiffrement ;

- **Le chiffrement symétrique** : Une seule et même clé sert à chiffrer et à déchiffrer les données (exemple : AES). Ce mode de chiffrement est très rapide et efficace pour chiffrer de grandes quantités de données, mais pose un problème de distribution sécurisée de la clé.

- **Le chiffrement asymétrique** : Il utilise une paire de clés, une publique pour chiffrer et une privée pour déchiffrer (exemple : RSA). Il résout le problème de distribution des clés, mais il est beaucoup plus lent, surtout pour chiffrer des volumes importants de données.

Le chiffrement RSA a été abordé dans la fiche n° 7 d'initiation à la cryptographie et le chiffrement AES dans l'une des fiches sur la cryptographie moderne.

2. Principe de fonctionnement du chiffrement hybride

Le chiffrement hybride combine ces deux méthodes pour tirer parti de leurs forces tout en minimisant leurs inconvénients. Le principe général de fonctionnement est le suivant :

- **Génération d'une clé de session symétrique** :

Le chiffreur génère une clé symétrique aléatoire (appelée **clé de session**), souvent à l'aide d'un algorithme. Cette clé est temporaire et utilisée uniquement pour une session spécifique.

- **Chiffrement des données avec la clé symétrique** :

La clé symétrique est utilisée pour chiffrer les données du message clair avec un algorithme de chiffrement symétrique rapide et efficace.

- **Chiffrement de la clé symétrique avec un algorithme asymétrique** :

La clé symétrique est ensuite chiffrée à l'aide d'un algorithme asymétrique (comme RSA) et de la clé publique du destinataire. Cette étape garantit que seule la personne possédant la clé privée correspondante peut déchiffrer la clé symétrique.

- **Transmission** :

Les données chiffrées du message et la clé symétrique chiffrée sont envoyées au destinataire.

- **Déchiffrement** :

Le destinataire déchiffre d'abord la clé symétrique avec sa clé privée de chiffrement asymétrique.

Il utilise ensuite cette clé symétrique pour déchiffrer le message.

3. Illustration pratique du chiffrement hybride avec Alice et Bob :

- Bob veut adresser un message à Alice

- Alice dispose d’une clé pour le chiffrement en RSA. Elle a donc diffusé publiquement N et son exposant public e. Bien entendu, elle garde toujours secret son exposant privé d.

Opérations à effectuer par Bob :

- Créer le message en clair.

- Créer une clé symétrique, appelée « clé de session ». Cette clé est généralement plus courte que le message clair.

- Chiffrer le message clair avec cette clé symétrique, par exemple avec un chiffrement AES.

- Chiffrer ensuite la clé symétrique par un chiffrement RSA avec la clé publique N et e d’Alice.

- Adresser à Alice en un seul fichier le message chiffré avec la clé symétrique et cette clé symétrique chiffrée en RSA avec la clé publique.

Opérations à effectuer par Alice :

Lorsqu’elle reçoit ce fichier, Alice déchiffre la clé symétrique grâce à sa clé privée d, puis déchiffre le message original avec cette clé symétrique.

4. Les avantages du chiffrement hybride

- **Efficacité :** L'algorithme symétrique est utilisé pour chiffrer les données volumineuses de manière rapide, tandis que l'algorithme asymétrique est utilisé uniquement pour protéger la clé de session, ce qui minimise l'impact de sa lenteur.

- **Sécurité de la clé :** L'utilisation d'un chiffrement asymétrique garantit que la clé symétrique ne peut être déchiffrée que par le destinataire prévu, ce qui résout le problème de distribution de clé dans le chiffrement symétrique.

- **Confidentialité :** Même si quelqu'un intercepte les données, il ne peut pas les déchiffrer sans la clé de session, qui elle-même est protégée par le chiffrement asymétrique

5. Les limites de ce type chiffrement

Bien que très efficace, le chiffrement hybride n'est pas exempt d'incertitudes :

Gestion des clés : Si une clé privée est compromise, toutes les sessions passées et futures pourraient être déchiffrées.

Attaques : Les attaques sur les algorithmes RSA ou d'autres algorithmes asymétriques peuvent compromettre la sécurité globale du chiffrement hybride. Cependant, en utilisant des tailles de clés suffisamment grandes et des protocoles de chiffrement modernes (comme les courbes elliptiques), ces attaques sont atténuées.

Un chiffre hybride actuel utilise le plus souvent un algorithme symétrique de type AES avec une clé de session de 128 symboles binaires (bits) qui permet de chiffrer de longs messages de manière sûre, associé à un RSA qui utilise des clés de 1024 à 2048 bits.

6. Les applications courantes du chiffrement hybride

Le chiffrement hybride est la méthode de base dans de nombreuses applications de sécurité, telles que :

- **SSL/TLS** : Utilisé pour sécuriser les connexions Internet entre un navigateur et un serveur web (par exemple, HTTPS).

- **PGP (Pretty Good Privacy)** : Utilisé pour le chiffrement des e-mails.

- **Messagerie sécurisée** : Des applications comme Signal ou WhatsApp utilisent le mode hybride pour échanger des messages.

*****

***

*