

# Module Cyber Sécurité

## Cryptographie Chiffrement symétrique

Jean-Marc MULLER

Sébastien SCHMITT

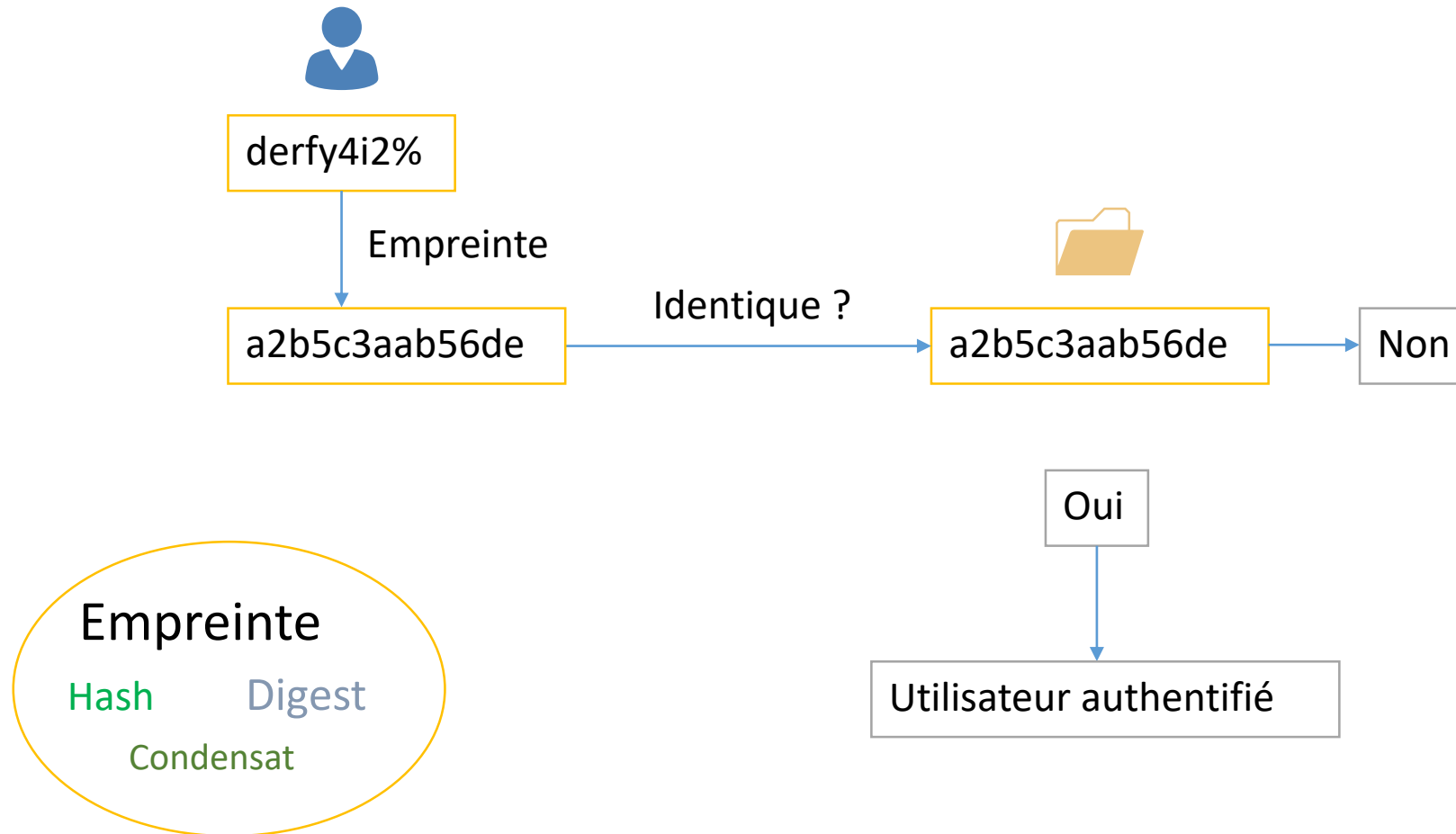


# Les besoins de chiffrement

- Terminologie "chiffrement"
  - ➔ Procédé cryptographique réversible permettant de rendre illisible et incompréhensible des données
  - ➔ Le terme "crypter" n'existe pas dans le dictionnaire (Anglicisme)
- Terminologie "Cryptographie" moderne
  - ➔ Ensemble des solutions de chiffrement qui assure la sécurité des données et en garantissant :
    - ➔ **Confidentialité** : rendre l'information incompréhensible à toute personne ne partageant pas le secret
    - ➔ **Intégrité** : garantir qu'une information reçue n'a pas été modifiée entre son émission et sa réception
    - ➔ **Authenticité (non répudiation)** : prouver de manière infalsifiable l'origine de l'information

# Le besoin d'intégrité

- Exemple d'usage fonction HASH (Gestion d'un mot de passe)



# Le besoin d'intégrité

- Fonction de "hash"
  - ➔ Renvoie un résultat formaté (Taille fixe)
  - ➔ Différent d'une fonction de codage (BASE64, ASCII ....)
  - ➔ Non réversible (fonction à sens unique)
  - ➔ Calcul très rapide
  - ➔ Peut garantir l'intégrité des données selon l'algorithme
  - ➔ Algorithme public

CHECKSUM

MD5

SHA (1,2,3,256,512)

LM Hash

...

# Le besoin d'intégrité

- Attaque des fonctions de hash
  - ➡ Par dictionnaire
    - ➡ Base de données des mots de passes les plus courants
  - ➡ Par force brute
    - ➡ Génération incrémentale de toutes possibilités
  - ➡ Par « Table arc-en-ciel »
    - ➡ Base de données en ligne disposant des toutes les paires (Mot de passe et condensat) possible selon les algorithmes

# Le besoin d'intégrité

- Protection des fonctions de hash

- ➔ Problèmes :

- ➔ Un même mot de passe génère un condensat identique
    - ➔ Il peut exister une collision mathématique (condensat identique)

- ➔ Solutions :

- ➔ Augmentation du temps (Délai après tentatives)
    - ➔ Augmenter la complexité de l'algorithme
    - ➔ Ajouter un élément aléatoire dans la fonction (Sel)

# Le besoin d'intégrité

- Fonction de salage

➡ Principe général :

Condensat =  $h(\text{mot de passe} + \text{sel})$

Digest =  $h(\text{password} + \text{nonce})$

+ est une opération logique ou mathématique

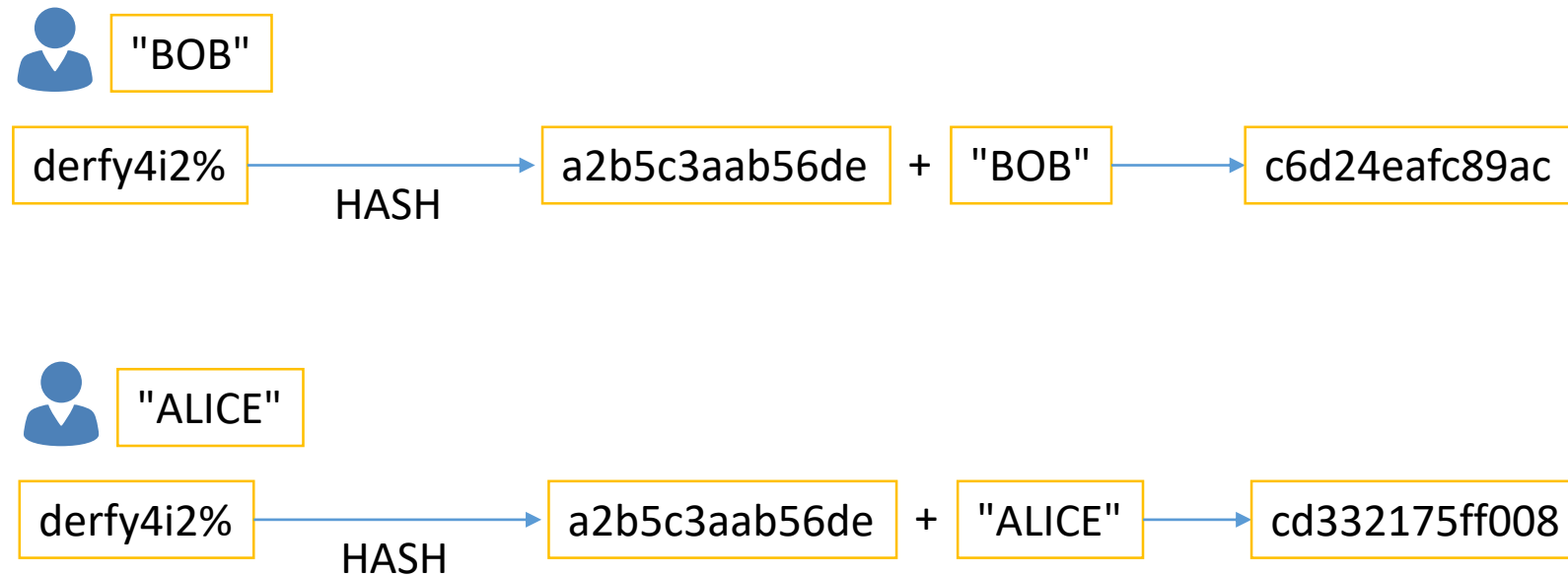
➡ Exemple :

- ➡ Concaténation du login et du mot de passe
- ➡ Multiplication avec la position de la souris
- ➡ NAND avec un fragment de mémoire

# Le besoin d'intégrité

- Fonction de salage

→ Exemple :

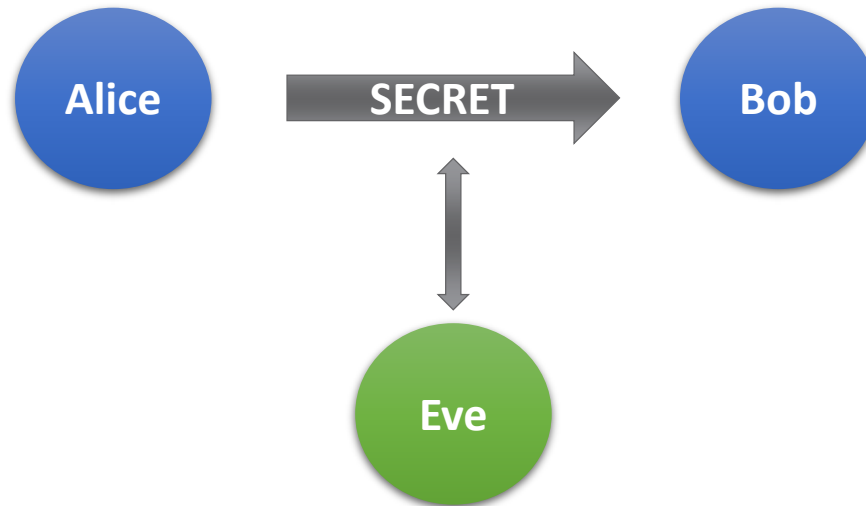




# Le besoin de confidentialité

- Modélisation

→ Alice veut transmettre un message secret à Bob, **comment faire ?**



# Le besoin de confidentialité

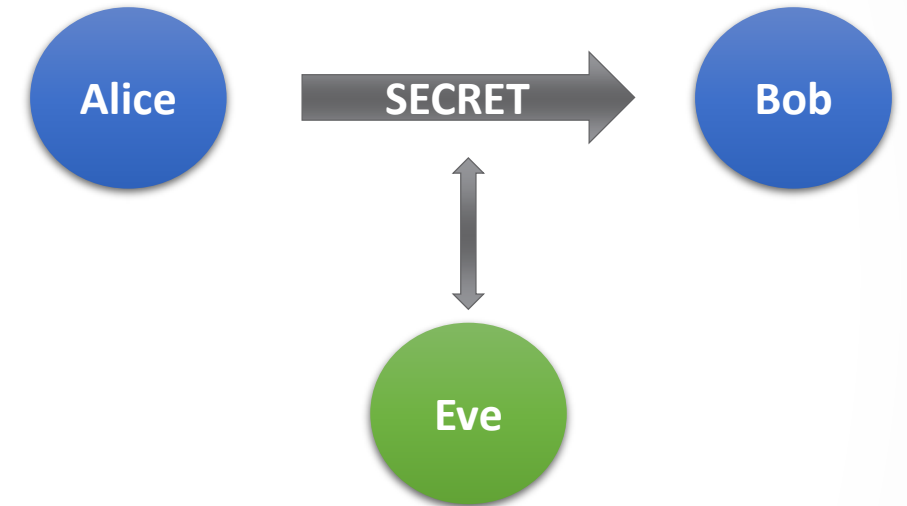
- Modélisation

➡ Alice veut transmettre un message secret à Bob

➡ **Alice** "chiffre" un message

➡ **Bob** reçoit le message et le "déchiffre"

➡ **Eve** intercepte le message



# Le besoin de confidentialité

- Chiffrement avec clé

- Symétrique

- César, Vigenère, Enigma, Chiffre de Vernam, DES, AES...



- Asymétrique

- DSA, EKE, ElGamal, RSA ...

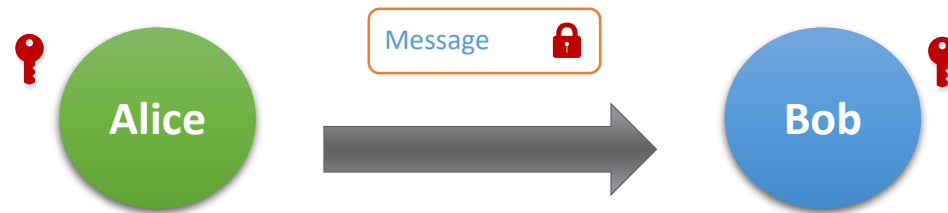


# Le besoin de confidentialité

- Cryptographie à clé secrète (partagé)

→ Alice chiffre un message avec sa clé et envoie celui-ci

→ César, Vigenère, Enigma, Chiffre de Vernam, DES, AES...



→ Bob déchiffre le message avec sa clé et peut répondre



# Le besoin de confidentialité

- Partage de secret multiple

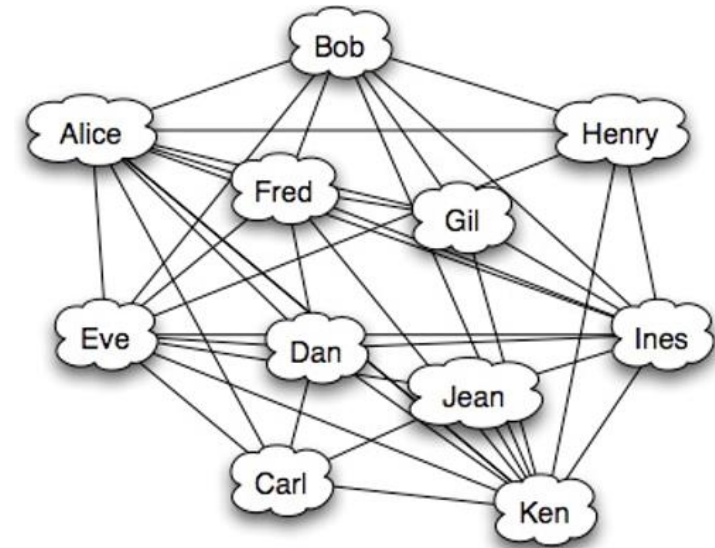
- ➔ Obligations

- ➔ La clé **K** doit être partagée en entre deux interlocuteurs
- ➔ Tous ces éléments doivent être absolument **SECRETS**

- ➔ Problématique

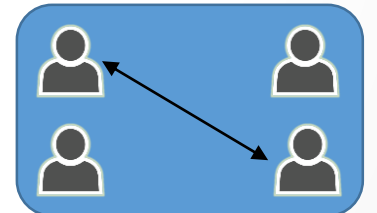
- ➔ Échange de clés sécurisées
- ➔ Vecteur de communication

Nb personnes	Nb de clés
2	1
5	10
100	4450
n	$n(n-1)/2$



## EXERCICE

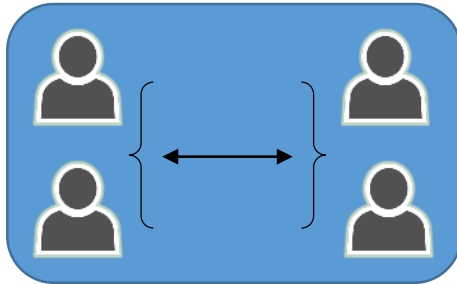
- 20 groupes de 4 personnes souhaitent utiliser un chiffrement symétrique pour discuter en secret selon les règles suivantes :
  - 10 groupes (type A) autorisent que toutes les personnes d'un même groupe partagent leurs secrets
  - 5 groupes (type B) n'autorisent pas que toutes les personnes d'un même groupe partagent leurs secrets
  - 5 groupes (type C) autorisent que la moitié des personnes d'un même groupe partagent leurs secrets entre eux
  - Un vecteur de communication contient une clé unique secrète
- Modéliser par des flèches les vecteurs de communication interne par type de groupe (sachant qu'un vecteur contient un clé unique secrète)
- Combien de clés internes sont nécessaires par type de groupe ?
- Combien de clés sont nécessaires pour la communication entre groupes ?
- Combien de clés au total sont nécessaires pour l'ensemble des communications ?



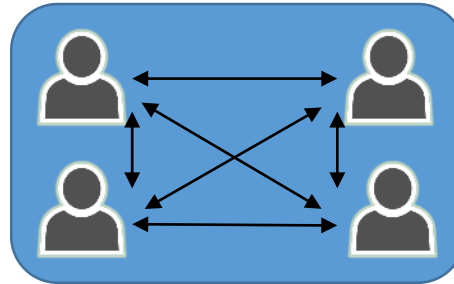
## CORRECTION

- ➔ Modéliser par des flèches les vecteurs de communication par type de groupe (sachant qu'un vecteur contient un clé unique secrète)

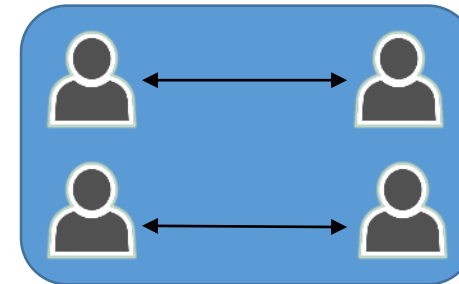
Groupe Type A



Groupe Type B



Groupe Type C



- ➔ Combien de clés sont nécessaires par type de groupe

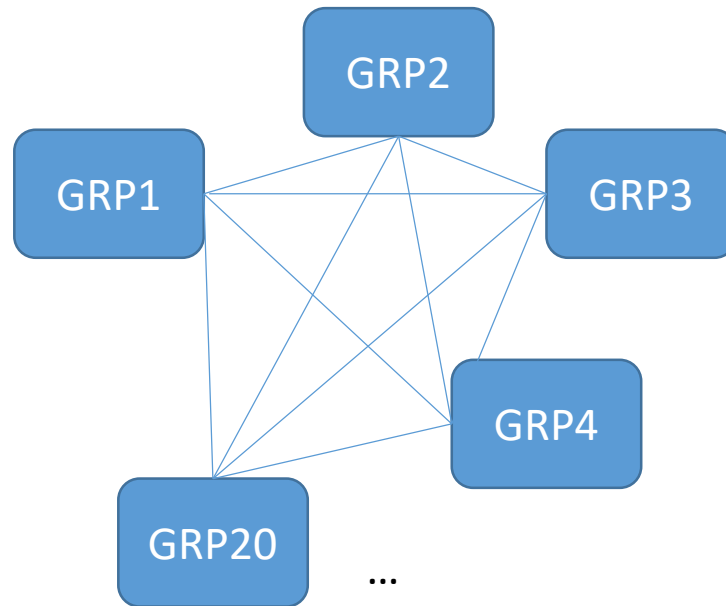
Type A = 1

Type B =  $4(4-1)/2 = 6$

Type C = 2

## CORRECTION

➡ Combien de clés sont nécessaires pour la communication entre groupes



Nbr de clés :

$$20(20-1)/2 = 190$$



## CORRECTION

➡ Combien de clés totales sont nécessaires

- ➡ 190 clés pour la communication entre groupes
- ➡  $10 \times 1$  clés pour la communication des groupes de type A
- ➡  $5 \times 6$  clés pour la communication des groupes de type B
- ➡  $5 \times 2$  clés pour la communication des groupes de type C

Nombre de clés totales :

**240**

# Le besoin de confidentialité

- Catégories de chiffrement symétrique
  - ➔ Chiffrement par bloc
    - ➔ Chaque élément est découpé en bloc de taille définie
    - ➔ Difficulté augmente avec la taille de la clé
    - ➔ Utilise d'opérations simples (Permutation, Substitution, Séparation...)
  - ➔ Chiffrement par flux (Stream Cipher)
    - ➔ Chaque élément est chiffré à la volée
    - ➔ Opérations simples et en nombre très limité
    - ➔ Surtout utilisé dans les équipements contraints (Téléphonie et sans fil)
    - ➔ Exemple : A5/1, RC4, E0 etc...

# Le besoin de confidentialité

- Chiffrement par flux

- Principe de base

- Utilisation de la fonction logique  $\oplus$  (XOR) et sa propriété involutive

XOR		
A	B	D
0	0	0
1	0	1
0	1	1
1	1	0

$$D = A \oplus B$$

$$D = (A \wedge \neg B) \vee (\neg A \wedge B)$$

$$D = (A \cdot \overline{B}) + (\overline{A} \cdot B)$$

- Générateur binaire pseudo-aléatoire

$$Gx = \{0,1\}^x$$

# Le besoin de confidentialité

- Chiffrement par flux

- Chiffrement

$$Mc = Md \oplus K(Gx)$$

*Mc* = Message Chiffré

*Md* = Message Déchiffré

- Déchiffrement

$$Md = Mc \oplus K(Gx)$$

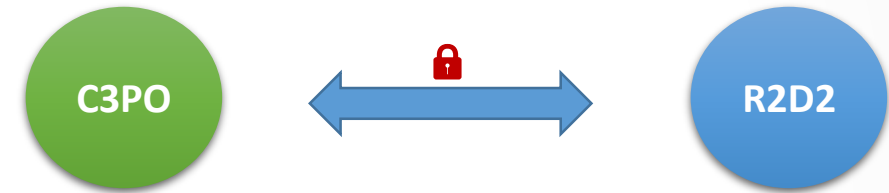
- Exemple

Md	0	0	1	0	1	1	1	0	1	0	1	0	1
K	1	0	0	0	1	1	0	1	1	0	1	0	1
Mc = Md $\oplus$ K	1	0	1	0	0	0	1	1	0	0	0	0	0

## EXERCICE

- Chiffrement par flux

→ C3PO souhaite discuter en permanence de manière "sécurisée" avec R2D2



→ Se mettent d'accord sur une clé secrète

→  $k1 = 0110\ 1110$

→ C3PO veut envoyer le message  $Md1 = 1100\ 1111$

→ Déterminer le message chiffré binaire noté  $Mc1$  en utilisant la clé  $k1$

→ R2D2 reçoit un nouveau message en hexadécimal  $Mc2 = 51\ 6C$

→ Quel est sont les caractères ASCII déchiffrés notés  $Md2$

→ R2D2 a compris que  $Md2$  est la nouvelle clé secrète notée  $k2$  et il veut répondre  $Md3 = R2D2$

→ Quel est le message chiffré  $Mc3$  en hexadécimal réceptionné par C3PO

# ASCII TABLE

## TD CRYPTOGRAPHIE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

## CORRECTION

- Chiffrement par flux

➔ Déterminer le message chiffré binaire noté Mc1 en utilisant la clé k1

Md1	1	1	0	0	1	1	1	1
K1	0	0	1	0	0	1	0	0
Mc1 = Md $\oplus$ K1	1	1	1	0	1	0	1	1

➔ Quels sont les caractères ASCII déchiffrés notés Md2

Mc2 (HEX)	51								6C							
Mc2 (DEC)	81								108							
Mc2 (BIN)	0	1	0	1	0	0	0	1	0	1	1	0	1	1	0	0
K1	0	0	1	0	0	1	0	0	0	0	1	0	0	1	0	0
Md2 = Mc2 $\oplus$ K1	0	1	1	1	0	1	0	1	0	1	0	0	1	0	0	0
Md2 (DEC)	117								72							
Md2 (ASCII)	u								H							

## CORRECTION

- Chiffrement par flux

➔ Quel est le message chiffré Mc3 en hexadécimal réceptionné par C3PO

Mc3 (ASCII)	R								2								D								2									
Mc3 (DEC)	82								50								68								50									
Mc3 (BIN)	0	1	0	1	0	0	1	0	0	0	1	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	1	1	0	0	1	0	
K2	0	1	1	1	0	1	0	1	0	1	0	0	0	1	0	0	0	1	1	1	0	1	0	1	0	1	0	1	0	0	0	1	0	0
Md3 = Mc3 ⊕ K2	0	0	1	0	0	1	1	1	0	1	1	1	0	1	1	0	0	0	1	1	0	0	0	1	0	1	1	1	1	0	1	1	0	
Mc3 (DEC)	39								118								49								118									
Mc3 (HEX)	27								76								31								76									



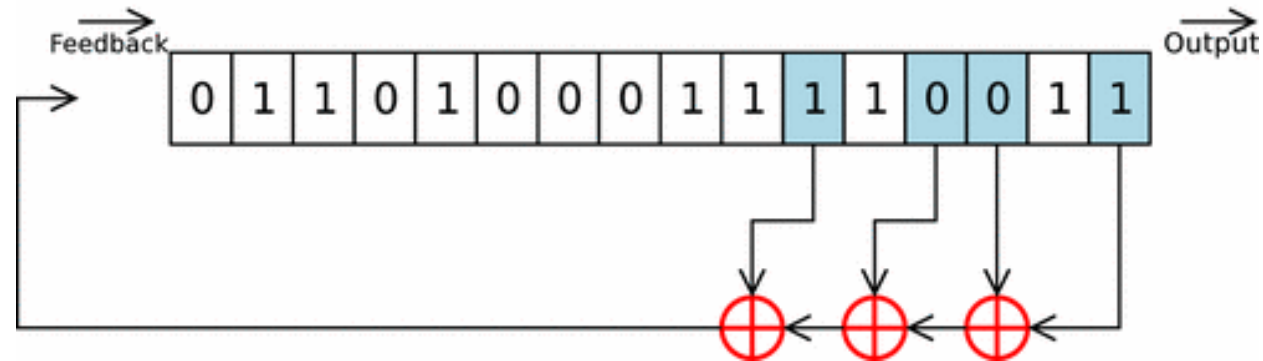
# Le besoin de confidentialité

- Chiffrement par flux (Générateur pseudo-aléatoire)
  - ➔ Registre à décalage à rétroaction linéaire (Linear Feedback Shift Register)
    - ➔ Système électronique ou logiciel
    - ➔ Il produit une suite de bits sous forme de registre à n-bits
    - ➔ Coût très faible (Matériel ou en calcul)
    - ➔ Utilisé en chiffrement ou en électronique
  - ➔ Faiblesse
    - ➔ Aléas généré par une solution prédictible et reproductible
    - ➔ Vulnérable aux attaques mathématique (Berlekamp-Massey)
    - ➔ Ne doit pas être utilisé seul (souvent en combinatoire)

# Le besoin de confidentialité

- Registre à décalage à rétroaction linéaire (LFSR)  $Gx = \{0,1\}^3$

➔ Modèle simple:

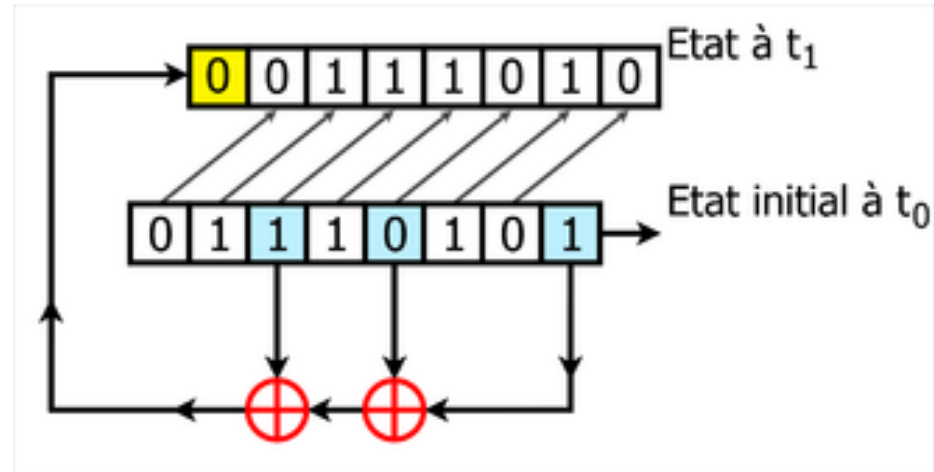


- ➔ Registre à décalage de n-bits
- ➔ D'une ou plusieurs fonctions linéaire (XOR dans notre cas)
- ➔ Réinjection du bit "calculé" dans le registre

# Le besoin de confidentialité

- Exemple Registre à décalage à réaction linéaire (LFSR)

→ Principe



Horloge	0	1	2	3	4	5	6	7	n
Registre 4 bits	01110101	00111010	00011101	00001110	10000111	11000011	11100001	01110000	11110000
Décimal	117	122	29	14	135	195	225	112	240

# Le besoin de confidentialité

- Chiffrement par flux A5/1 :

- Description

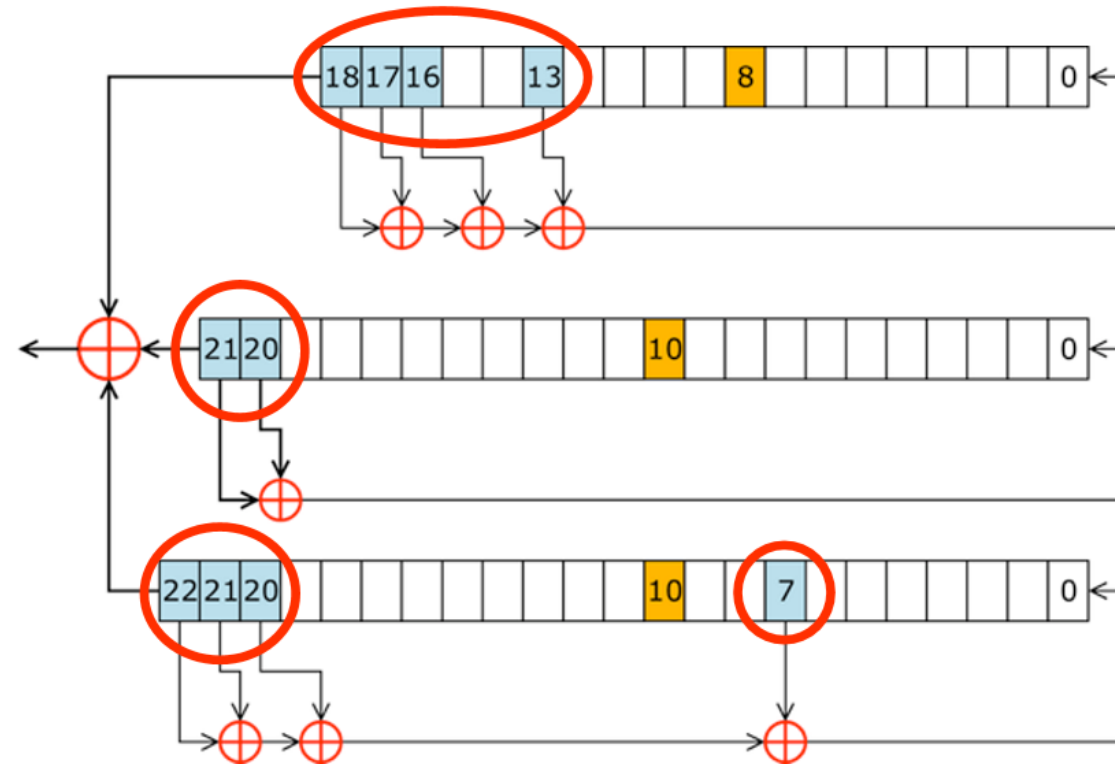
- Utilisé dans la communication GSM depuis 1994
- Chiffrement cassé en 2009
- La communication utilise deux flux (montant et descendant)

- Principe général

- Chaque flux transmet 114 bits chiffré dans un bloc de 4,6 ms
- Un générateur pseudo-aléatoire produit 228 bits
- Utilise trois registres à décalage à rétroaction linéaire
- Utilise une clé de session K entre le mobile et l'antenne

# Le besoin de confidentialité

- Générateur pseudo aléatoire (Protocole A5/1)



- ➔ 3 registres LFSR à position fixe
- ➔ XOR des 3 sorties de registre

# Le besoin de confidentialité

- Chiffrement par flux A5/1 (Attaques connues):
  - ➔ 2000 : Shamir
    - ➔ Possible de retrouver la clé si 2 minutes de communication en clair
    - ➔ Calcul de 248 étapes (soit plusieurs centaines de Go de données...)
  - ➔ 2003 : Ekdahl et Johansson
    - ➔ Attaque sur l'initialisation du générateur
    - ➔ 5mn de conversation en clair suffisent...
    - ➔ Pas besoin de grandes tables précalculées
  - ➔ 2009 : Chaos Computer Club
    - ➔ Déchiffrement en temps réel

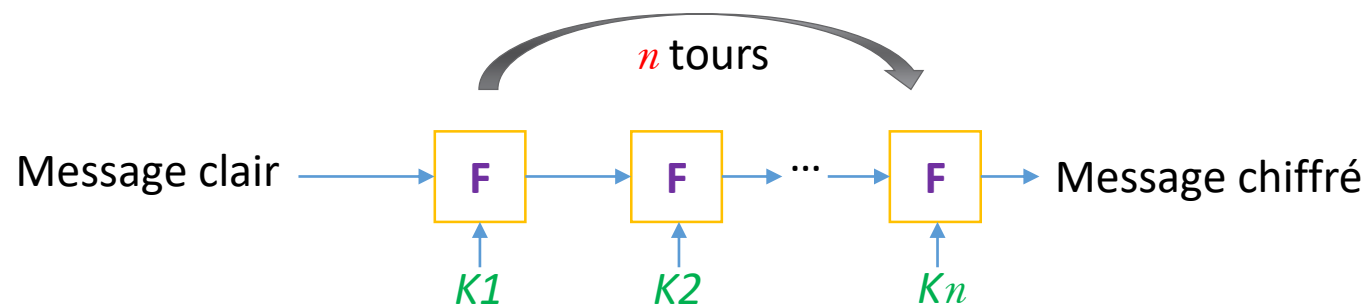
# Le besoin de confidentialité

- Chiffrement par bloc

- Description

- Découpage du texte clair en **blocs de même taille**
- Chiffrement itératif (*n* tours) par bloc
- La longueur des clés **K** doit  $> 128$  bits

- Principe général



# Le besoin de confidentialité

- Chiffrement par bloc

- Origine

- Horst Feistel, cryptologue d'IBM (1915 – 1990)
- A l'origine de la théorie du chiffrement par bloc
- Architecture similaire entre chiffrement et déchiffrement



- Réseau de Feistel

- Chiffrement en plusieurs étapes (tours, étages, rounds)
- Utilisation de principe simple :
  - La permutation linéaire (P-Box)
  - La substitution non linéaire (S-Box)
  - $\oplus$  (XOR)
  - Clé intermédiaire à chaque tour



# Le besoin de confidentialité

- Réseau de Feistel :

- ➔ Chiffrement

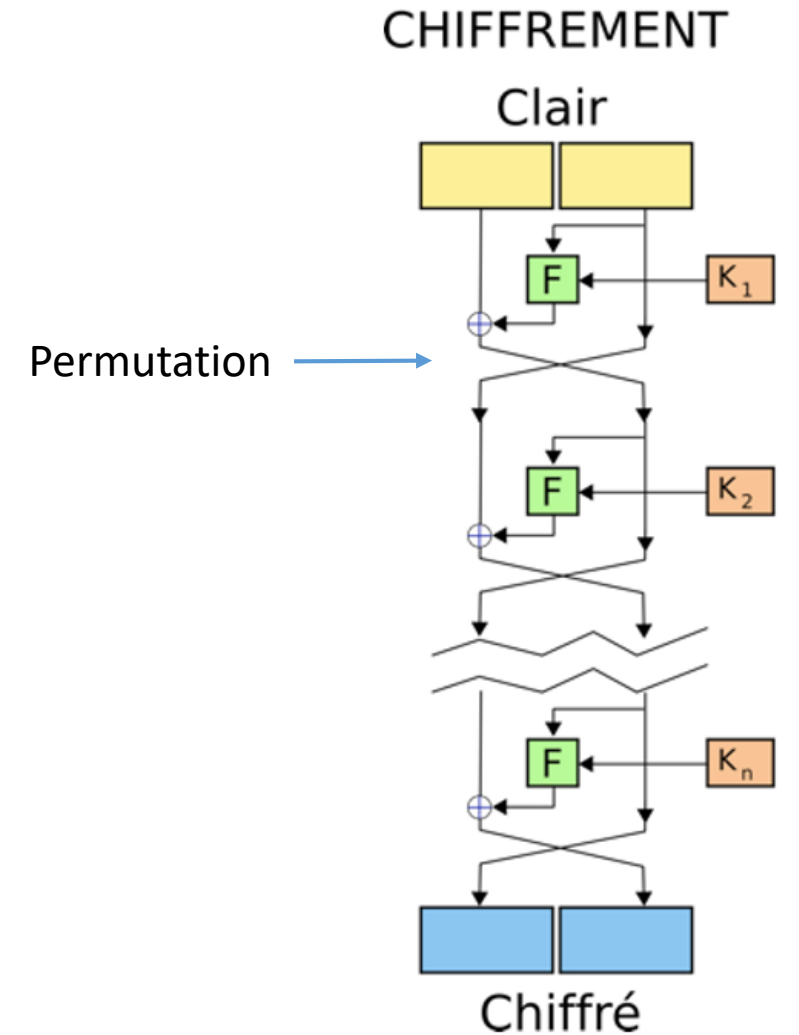
- ➔ Découpe bloc en 2 parties, **(L0 , R0)**
- ➔ A chaque tour  $i = 0, 1, \dots, n$

$$L_{i+1} = R_i$$

$$R_{i+1} = L_i \oplus F(R_i, K_i)$$

- ➔ Le message chiffré est :

$$R_{n+1}, L_{n+1}$$



# Le besoin de confidentialité

- Réseau de Feistel :

- ➔ Chiffrement

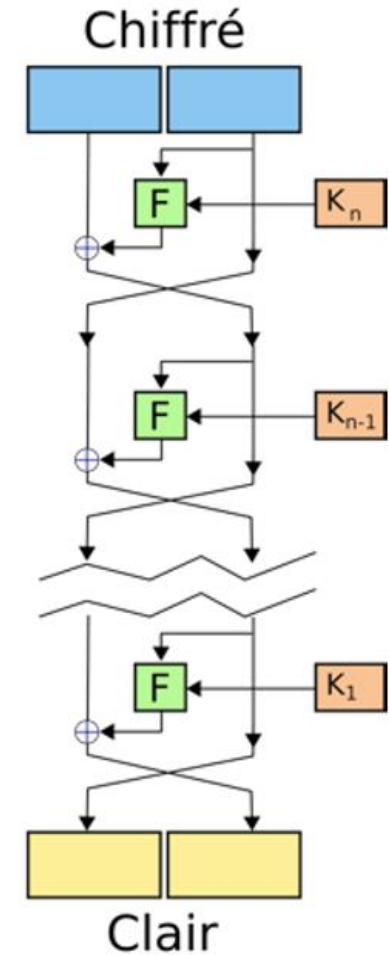
- ➔ Bloc chiffré ( $R_{n+1} = L_{n+1}$ )
    - ➔ A chaque tour  $i = n, n-1, \dots, 0$

$$R_i = L_{i+1}$$

$$L_i = R_{i+1} \oplus F(L_{i+1}, K_i)$$

- ➔ Le message chiffré est :

$$L_0, R_0$$



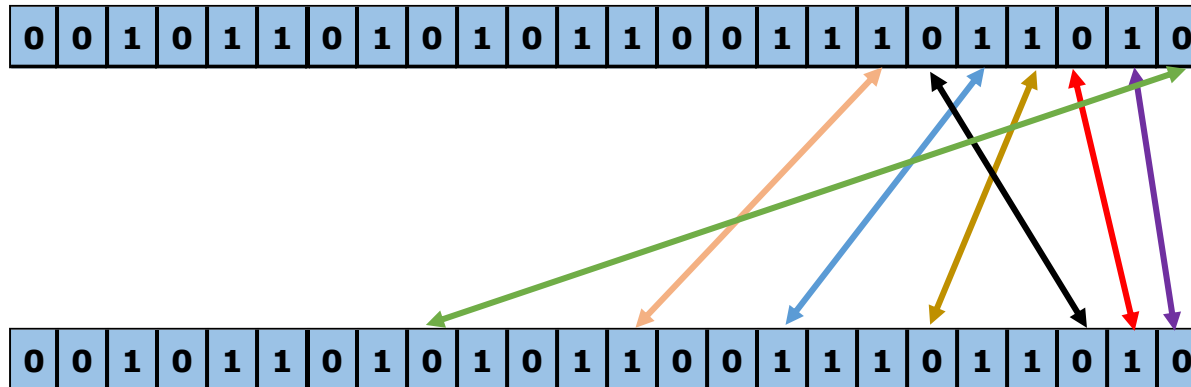
# Le besoin de confidentialité

- Chiffrement par bloc

- ➔ Principe de la P-Box

- ➔ Permutation successive des bits selon une matrice

Matrice P-Box : [16,0,1,6,9,3,12 ....]



# Le besoin de confidentialité

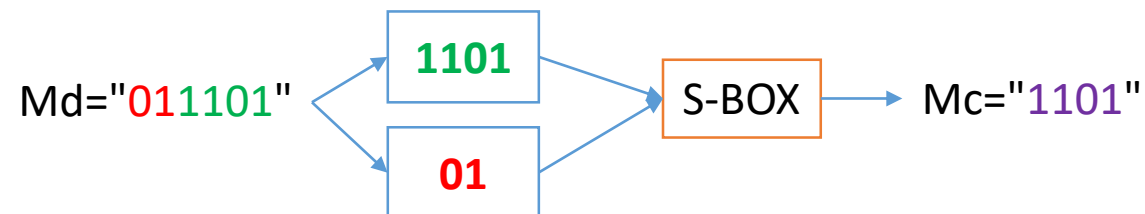
- Chiffrement par bloc

- ➔ Principe de la S-Box

- ➔ Identique au code de César (Substitution)
- ➔ Peut-être aussi une fonction de compression
- ➔ Exemple de S-BOX avec compression 6/4

S <sub>5</sub>		Middle 4 bits of input															
		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Source : wikipedia.org



# Le besoin de confidentialité

- Chiffrement DES (Data Encryption Standard):

- ➔ Principe

- ➔ Chiffrement symétrique par bloc de 64 Bits (8 octet)
- ➔ Utilise des clés 58 Bits (6 bits pour le contrôle de parité)
- ➔ Comporte 16 itérations
- ➔ Basé sur le schéma de Feistel
- ➔ Datant de 1975, il est remplacé par le 3DES

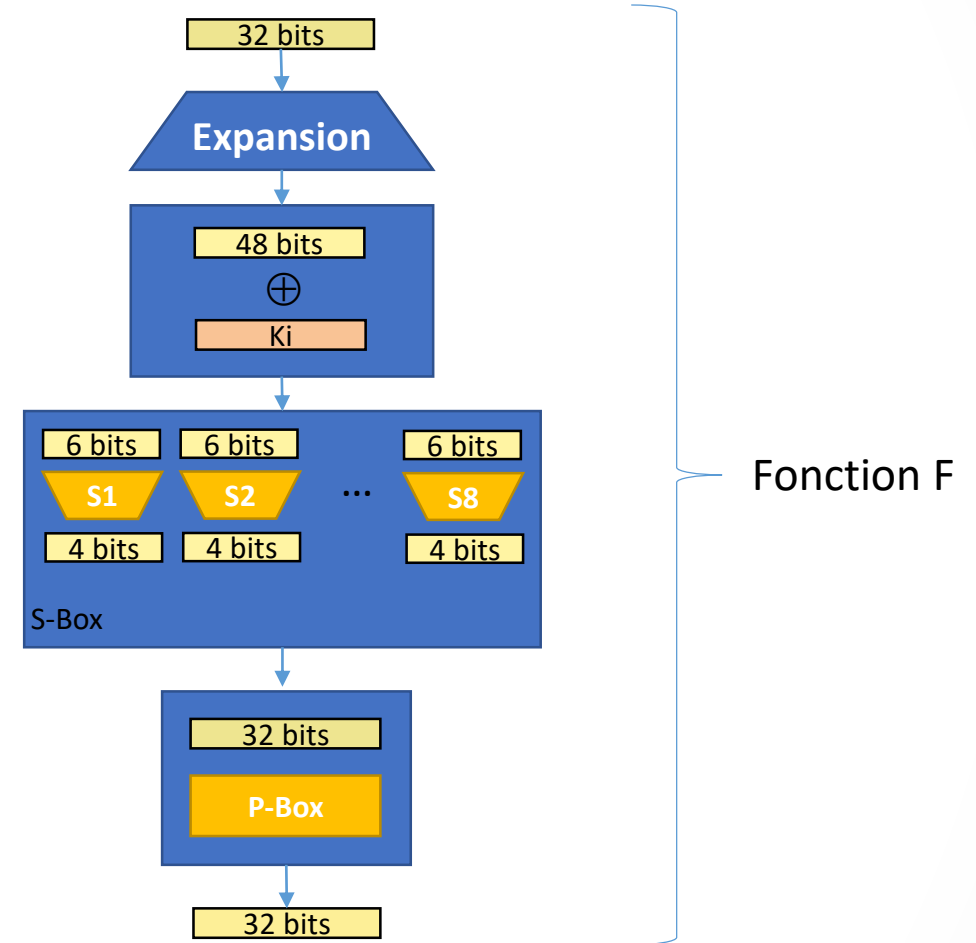
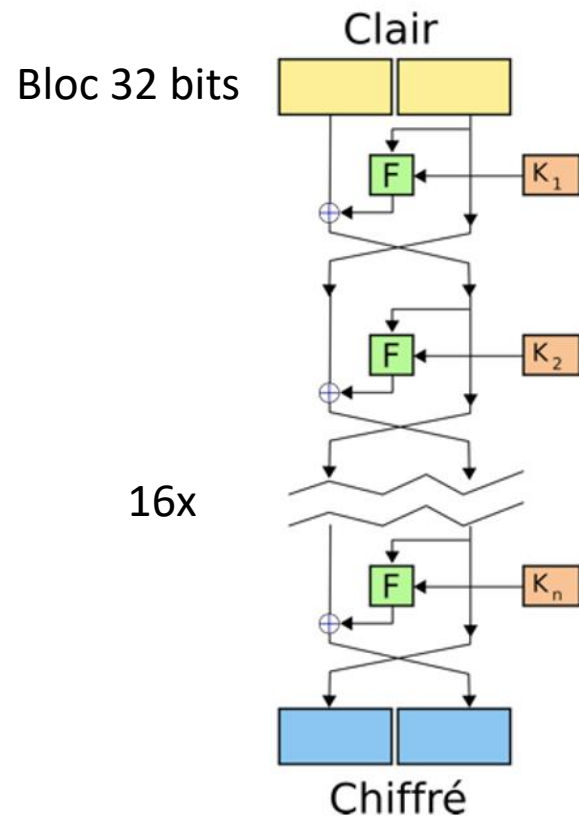
- ➔ Attaque

- ➔ 1980 : Pour un même texte, pré-calculer toutes les versions chiffrées
- ➔ 1983 : cryptanalyse linéaire (243 couples clair/chiffré avec la même clé)
- ➔ 1998 : Attaque par force brute (250 000 \$ et 56 h)

# Le besoin de confidentialité

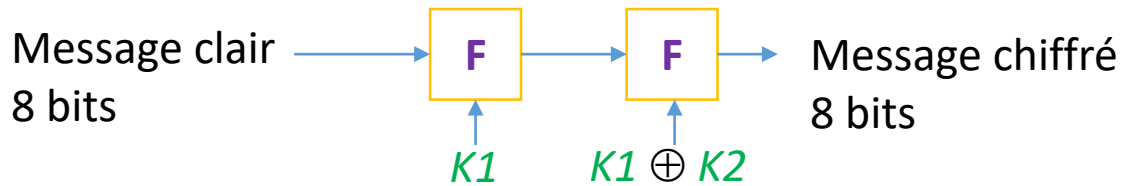
- Chiffrement DES (Data Encryption Standard):

## → Principe



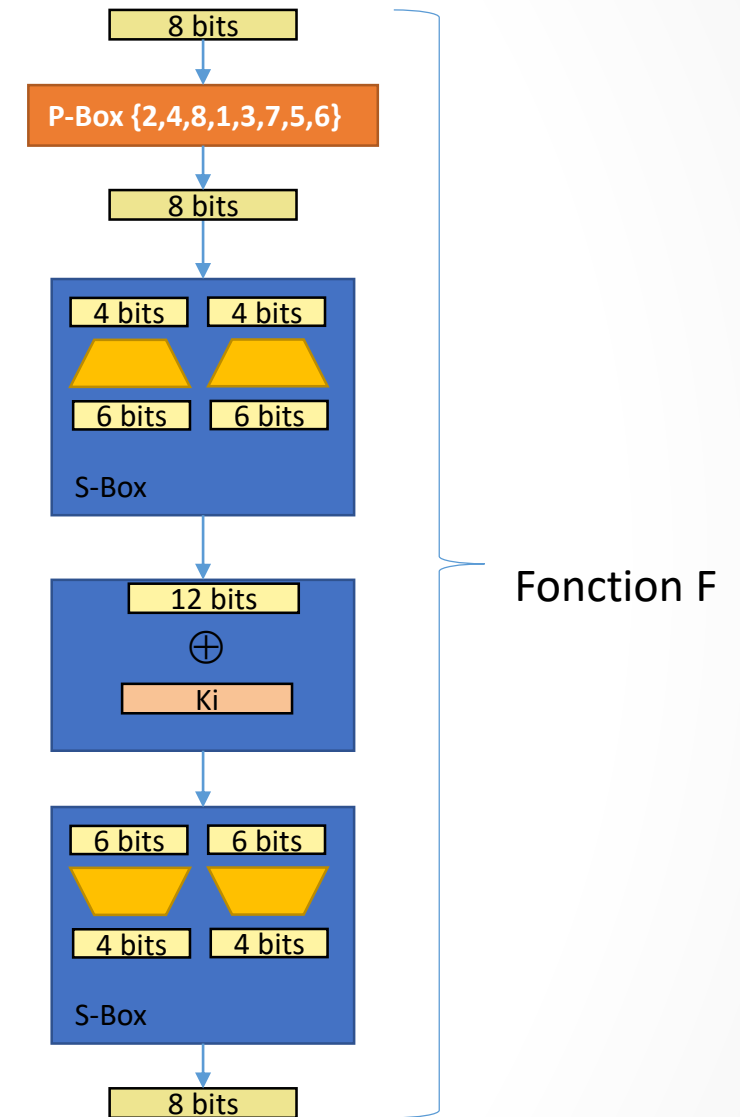
# EXERCICE

- Chiffrement symétrique
  - Soit l'algorithme de chiffrement ci-dessous
    - $K1$  (BIN) = 1100 0110 1010
    - $K2$  (DEC) = 952
    - Message clair (HEX) = 76



$S_5$		Matrice S-Box															
		Middle 4 bits of input															
Outer bits	00	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

→ Quel est le message chiffré ?



# TD CRYPTOGRAPHIE

- Chiffrement symétrique

→ 1<sup>er</sup> passage P-BOX {2,4,8,1,3,7,5,6}

Message	1	1	0	1	1	1	0	0
P-BOX	0	1	1	1	0	1	1	0

→ 1<sup>er</sup> passage S-BOX 4/6

P-BOX		0	1	1	1			0	1	1	0	
S-BOX	0	0	0	1	0	0	0	0	0	1	1	1

→ XOR avec K1

S-BOX	0	0	0	1	0	0	0	0	0	1	1	1
XOR K1	1	1	0	0	0	1	1	0	1	0	1	0
RESULTAT	1	1	0	1	0	1	1	0	1	1	0	1

→ 1<sup>er</sup> passage S-BOX 6/4

RESULTAT	1	1	0	1	0	1	1	0	1	1	0	1
S-BOX		1	1	1	0			0	0	1	1	



# TD CRYPTOGRAPHIE

- Chiffrement symétrique

→ 2<sup>ème</sup> passage P-BOX {2,4,8,1,3,7,5,6}

Message	1	1	1	0	0	0	1	1
P-BOX	1	0	1	0	1	1	0	1

→ 2<sup>ème</sup> passage S-BOX 4/6

P-BOX		1	0	1	0			1	1	0	1	
S-BOX	1	0	0	1	0	0	0	1	0	1	1	0

→ K1 XOR K2

K1	1	1	0	0	0	1	1	0	1	0	1	0
K2	0	0	1	1	1	0	1	1	1	0	0	0
K1 XOR K2	1	1	1	1	1	1	0	1	0	0	1	0

# TD CRYPTOGRAPHIE

- Chiffrement symétrique

→ XOR (K1 XOR K2)

S-BOX	1	0	0	1	0	0	0	1	0	1	1	0
K1 XOR K2	1	1	1	1	1	1	0	1	0	0	1	0
RESULTAT	0	1	1	0	1	1	0	0	0	1	0	0

→ 2<sup>ème</sup> passage S-BOX 6/4

S-BOX	0	1	1	0	1	1	0	0	0	1	0	0
MESSAGE		1	0	1	0			0	1	1	1	

# Chiffrement symétrique


- Chiffrement AES (Advanced Encryption Standard):

- ➔ Principe

- ➔ Algorithme (aussi nommé Rijndael) crée en 2000
- ➔ Utilise des blocs et des clés de 128 à 256 Bits
- ➔ Itération de respectivement 10,12 et 14 tours
- ➔ N'est pas basé sur un schéma de Feistel
- ➔ Considéré comme sûr à ce jour

- ➔ Attaque

- ➔ La seule attaque possible est la force brute
- ➔ Seul des versions simplifiées avec 8 tours ont été cassées
- ➔ Microsoft (2011) sur la version complète avec  $2^{126}$  au lieu de  $2^{128}$  opérations soit 1 milliard d'année au lieu de 5 milliards !!!

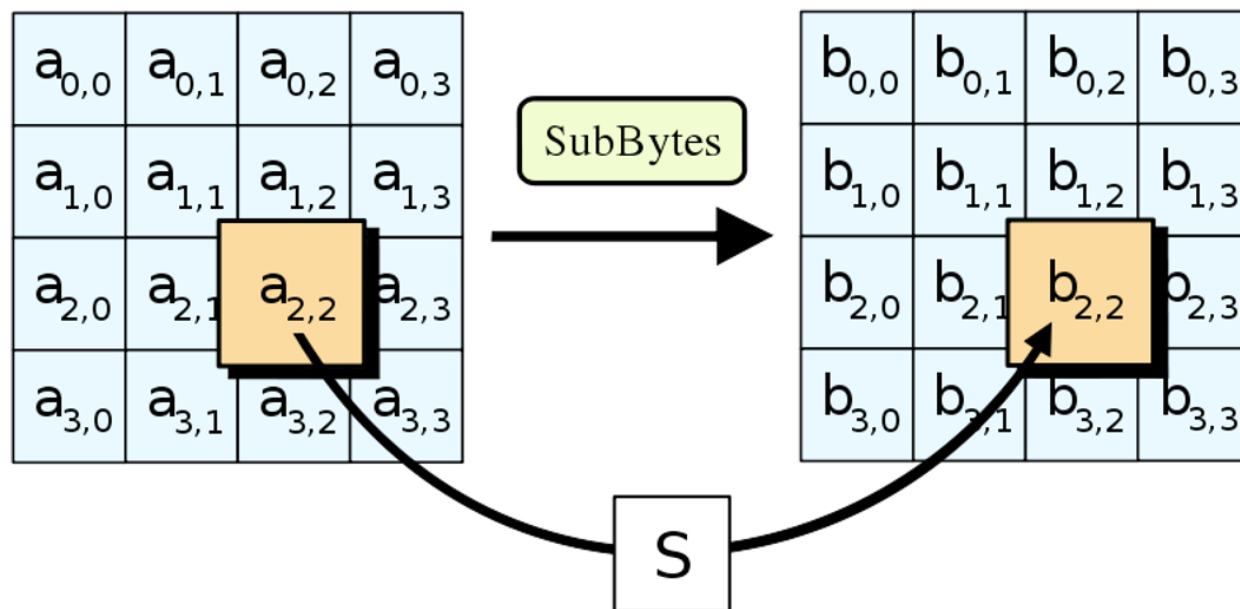


	Rijndael	Serpent	Twofish	MARS	RC6
General Security	2	3	3	3	2
Implementation Difficulty	3	3	2	1	1
Software Performance	3	1	1	2	2
Smart Card Performance	3	3	2	1	1
Hardware Performance	3	3	2	1	2
Design Features	2	1	3	2	1
Total	16	14	13	10	9

Concours année 2000

# Chiffrement symétrique

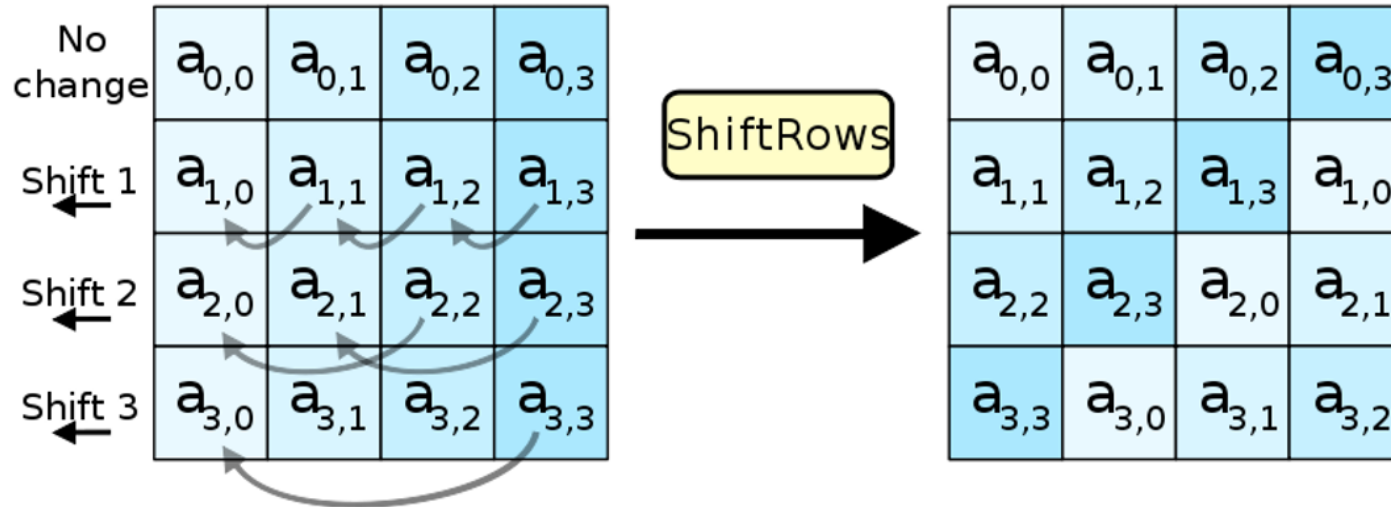
- SubBytes



➡ Substitution des 16 octets (128 bits selon une table prédéfinie)

# Chiffrement symétrique

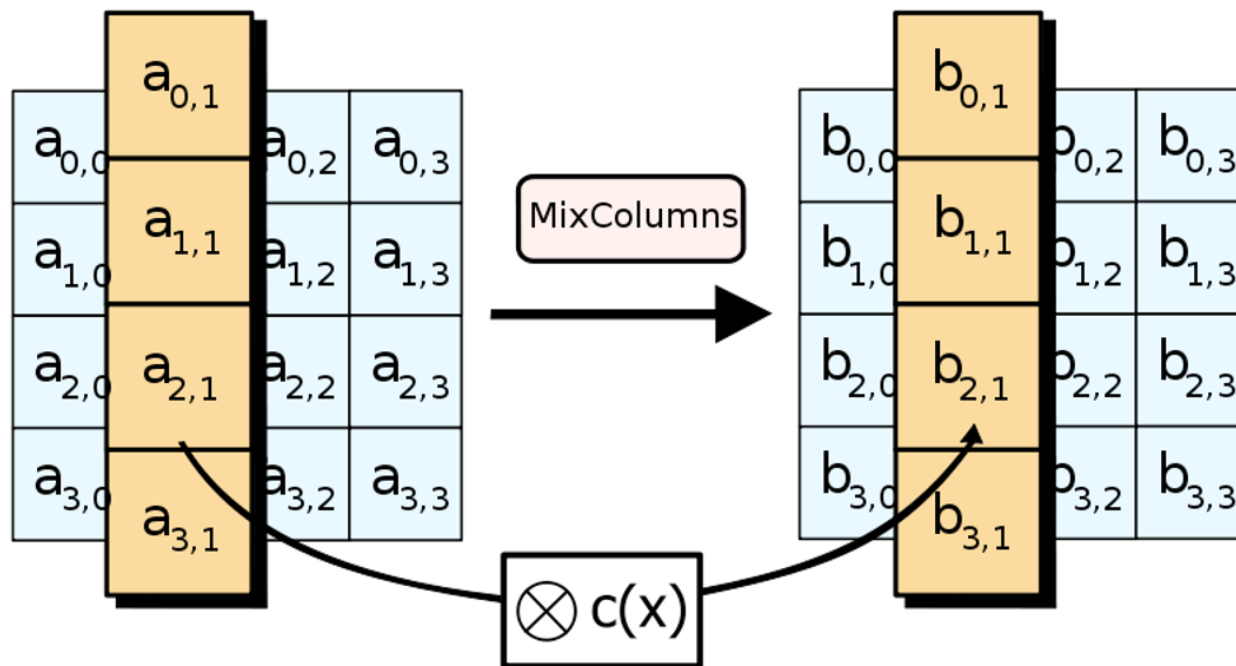
- ShiftRows



➡ Décalage variable des octets selon la ligne

# Chiffrement symétrique

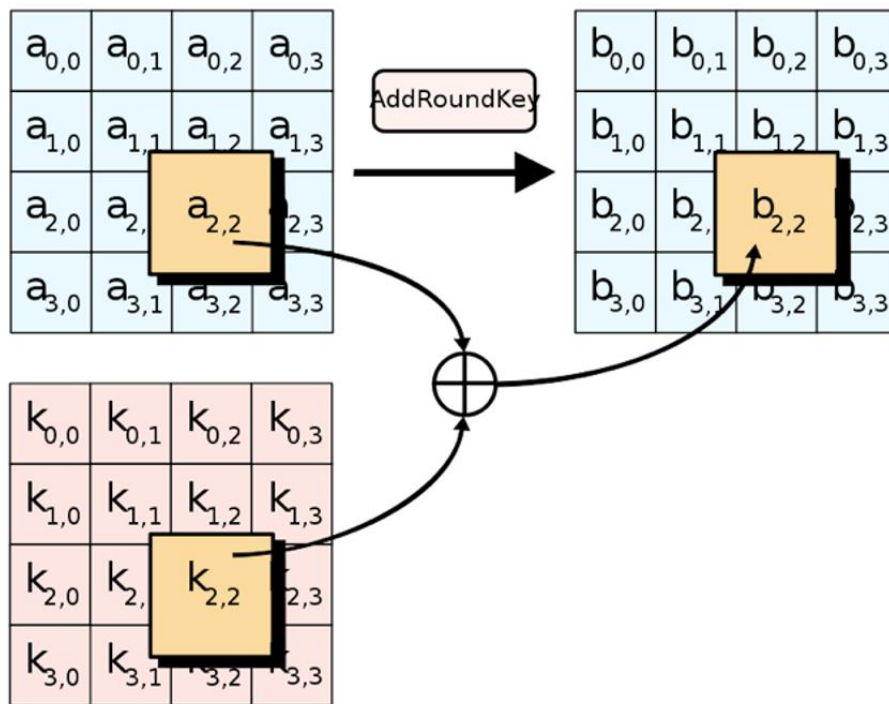
- MixColumns



➡ Multiplication de chaque colonne par une matrice prédéfinie

# Chiffrement symétrique

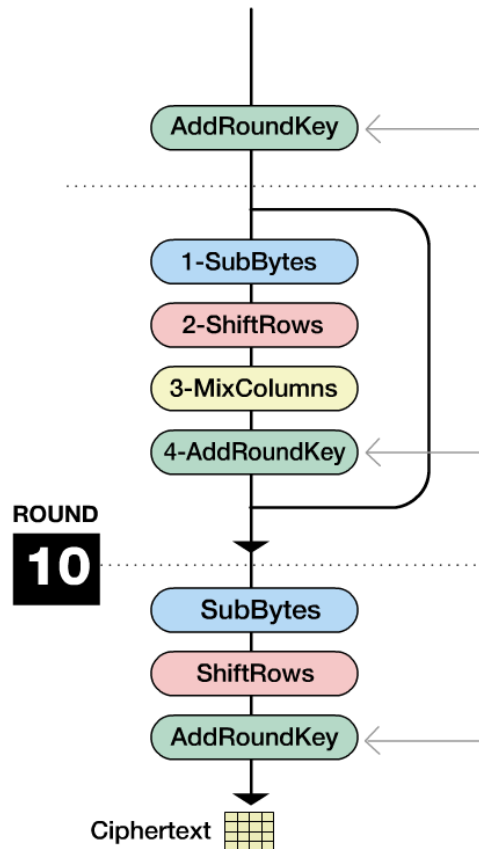
- AddRoundKey



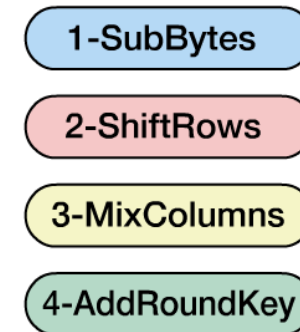
➡ Fonction XOR avec la clé

# Chiffrement symétrique

- Chiffrement AES 128



- 4 blocs de fonctions
- 10 itérations
- La clé de chaque itération est dérivé par fonction successive (X, S-Box,  $\oplus$ )





# Chiffrement symétrique

- Conclusion

- ➔ Nécessite un partage de secret
- ➔ Chiffrement très facile avec clé
- ➔ Déchiffrement très (très) difficile sans la clé
- ➔ Une clé par interlocuteur potentiel
- ➔ Utilise des algorithmes connus :
  - ➔ **DES** (Data Encryption Standard) : clé 56 bits
  - ➔ **3DES** : 3 passes dans DES avec 2 ou 3 clés distinctes (112 ou 168 bits)
  - ➔ **RC2, RC4, RC6** : clé de 1 à 1024 bit
  - ➔ **IDEA** (International Data Encryption Algorithm) : clé de 128 bits
  - ➔ **AES** (Advanced Encryption Standard) : clé de 128, 192, 256 bits
  - ➔ **BLOWFISH** : clé variable de 1 à 448 bits
  - ➔ ....

Source : <http://www.cs.bc.edu/~straubin/cs381-05/blockciphers/>

**FIN**