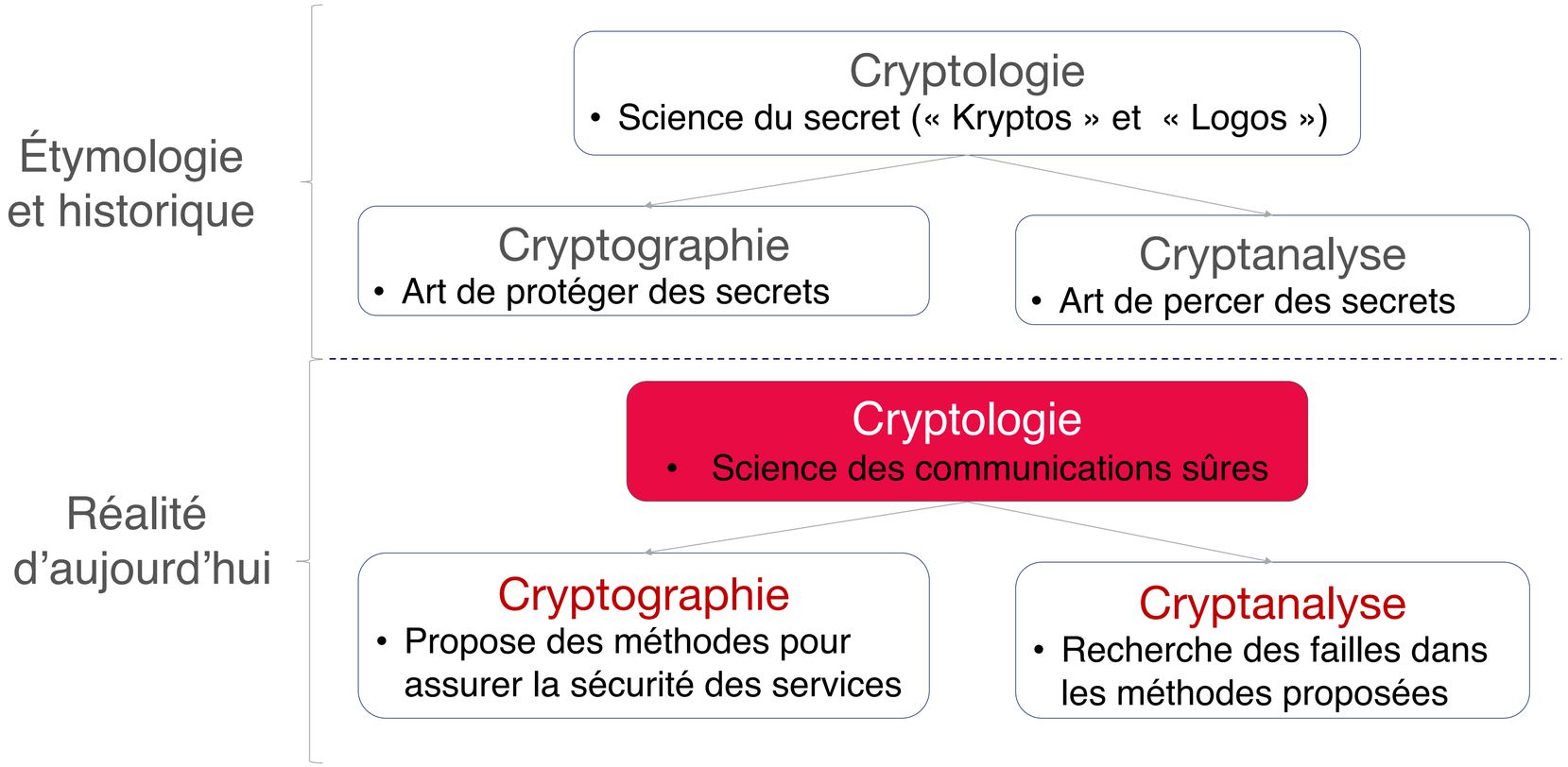
InterSem-INF114
Cryptographie

Sébastien Canard
Année 2023-2024

Définitions de base de la cryptologie



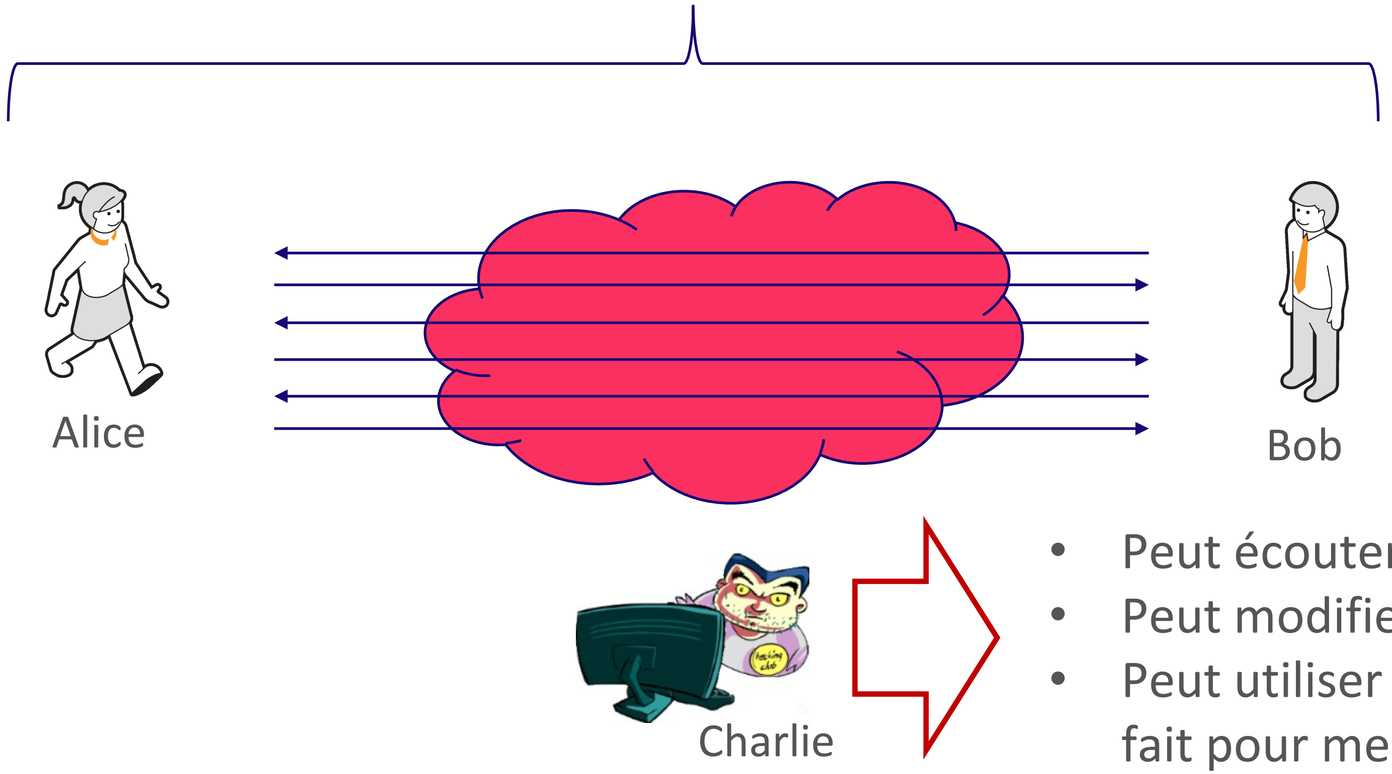
- La cryptographie est un ensemble de techniques applicables à l'échange et au traitement de l'information, permettant de garantir la sécurité des services
- Depuis les années 1970: discipline scientifique principalement basée sur les mathématiques (problèmes difficiles, analyse) et l'informatique (complexité, implémentation)

Services de sécurité

Service de sécurité	Outil cryptographique
• Confidentialité • Pas de divulgation d'information non autorisée • Seules les entités autorisées peuvent observer une information donnée	Chiffrement à clé secrète/publique
• Intégrité • Pas de modification d'information non autorisée • Seules les entités autorisées peuvent modifiée une information donnée	Signature, fonction de hachage, MAC
• Non-répudiation • Une entité ne doit pas, après coup, dénier des actions ou engagements antérieur	Signature
• Authentification • Processus pour vérifier l'identité d'une entité ⇒ authentification de personne • Processus pour vérifier la provenance d'une information donnée	Signature, MAC

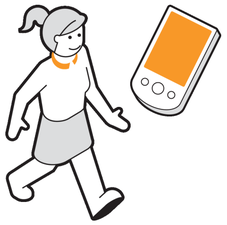
Voici Alice et Bob… Et Charlie

- Communication de beaucoup de données
- Confidentialité, authenticité, intégrité des données
- Non répudiation



- Peut écouter tous les échanges
- Peut modifier les éléments échangés
- Peut utiliser tout ce qu'il a écouté et fait pour mener une attaque

Quelques contextes



Alice

2G/3G/4G/5G

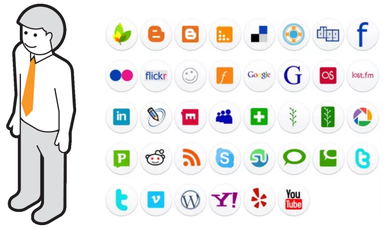
Bob

Charlie

- Écoute illicite
- Faux client

Alice

HTTPS (TLS)



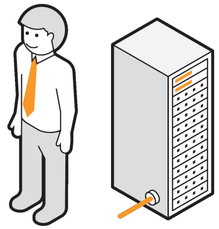
Bob

Charlie

- Écoute illicite
- Récupération de données (client et/ou serveur)

Alice

SSH

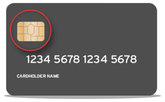


Bob

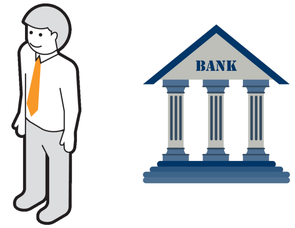
Charlie

- Écoute illicite
- Faux accès

Alice



EMV



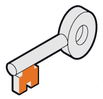
Bob

Charlie

- Faux accès

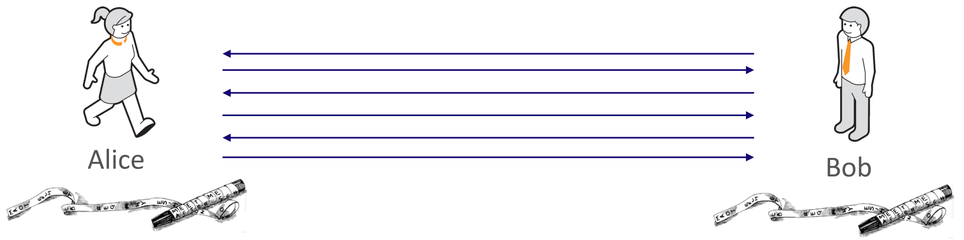
1. Commençons par la confidentialité

Méthode de permutation – Exemple de la Syctale

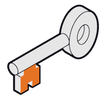


Principe : mélanger/permuter les lettres/mots d'un texte

- Principe de la Scytale (bâton de Plutarque)
  - Utilisé par les Spartiates (404 av. J.-C.)
  - Il faut pouvoir avoir un moyen facile d'exécuter la permutation inverse, sans qu'un ennemi ne puisse le faire facilement

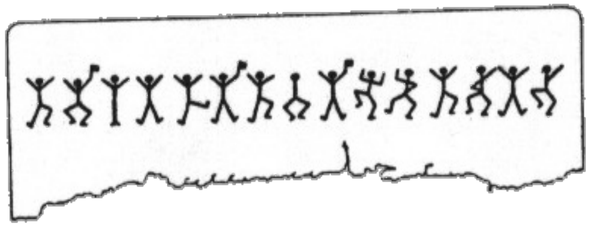


Méthode de substitution



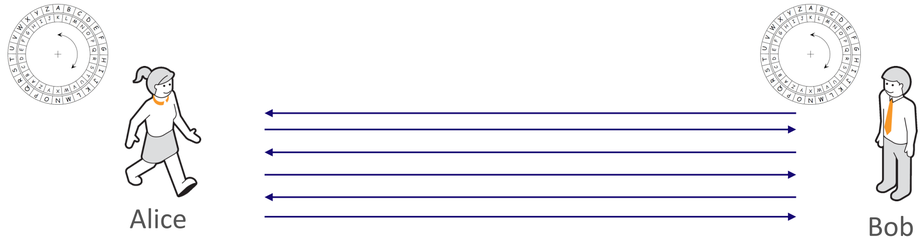
Principe : remplacer/substituer une lettre/mot par autre chose

- En utilisant un autre alphabet
  - Exemple des « Hommes dansants » (Sir Arthur Conan Doyle) avec Sherlock Holmes

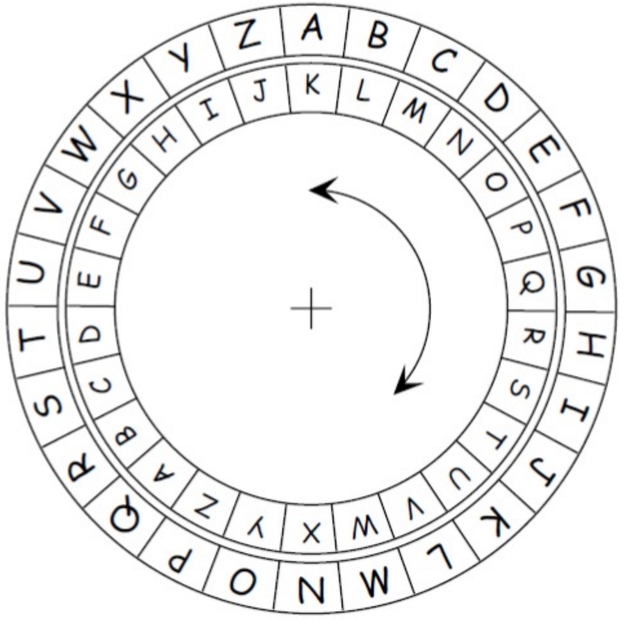


- En utilisant le même alphabet
  - Problématique : comment rapidement appliquer la substitution inverse sans donner de moyen facile pour l'ennemi de retrouver le texte initial

Système de César



- Utilisé au 1er siècle (par Jules César ?)
- Principe : simplifier la mémorisation de la substitution
  - Rotation de l'alphabet d'une certaine valeur (10 ⇒ A = K)
  - Remplacer chaque lettre par la valeur correspondante
  - Déchiffrement : lire la roue dans l'autre sens



- Exemple de texte clair

C	E	M	E	S	S	A	G	E	E	S	T	C	H	I	F	F	R	E	A	V	E	C	L	E	S	Y	S	T	E	M	E	D	E	C	E	S	A	R

- Et du texte chiffré correspondant

M	O	W	O	C	C	K	Q	O	O	C	D	M	R	S	P	P	B	O	K	F	O	M	V	O	C	I	C	D	O	W	O	N	O	M	O	C	K	B

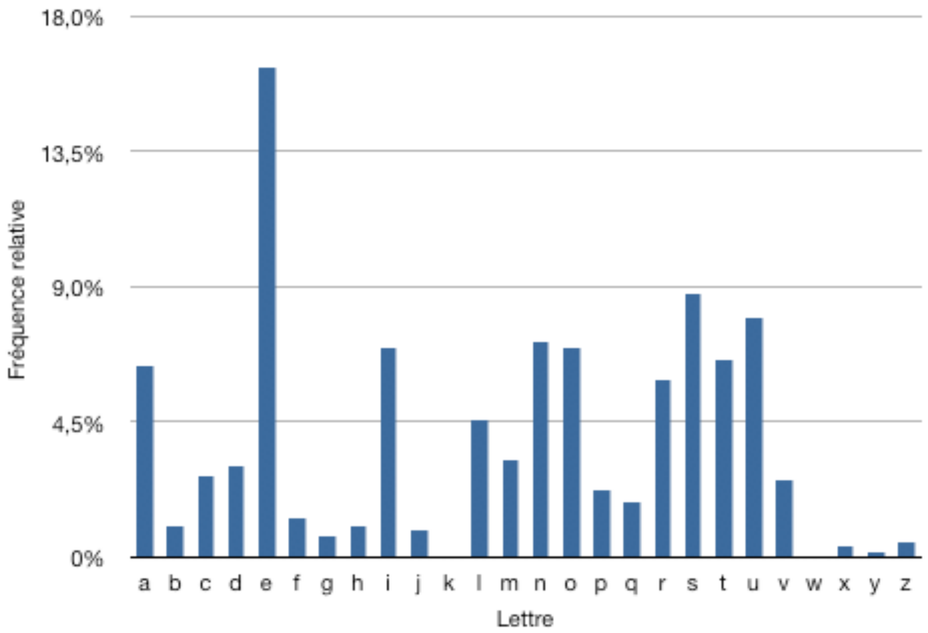
TD – Cryptanalyse de César

- Nombre de possibilités assez faible : 26
  - Tester toutes les combinaisons possible…
- Meilleure méthode formalisée par Al-Kindi au IXème siècle
- Exemple :

D	S	U	J	Q	H	L	G	D	G	Y	A	W	W	K	L	D	S	K	U	A	W	F	U	W	V
M	K	W	U	J	W	L	W	D	D	W	W	F	Y	D	G	T	W	D	S	U	J	Q	H	L	G
Y	J	S	H	Z	A	W	D	W	U	J	A	L	M	J	W	K	W	U	J	W	L	W	W	L	D
S	U	J	Q	H	L	S	F	S	D	Q	K	W	D	S	F	S	D	Q	K	W	V	W	U	W	L
L	W	V	W	J	F	A	W	J	W																

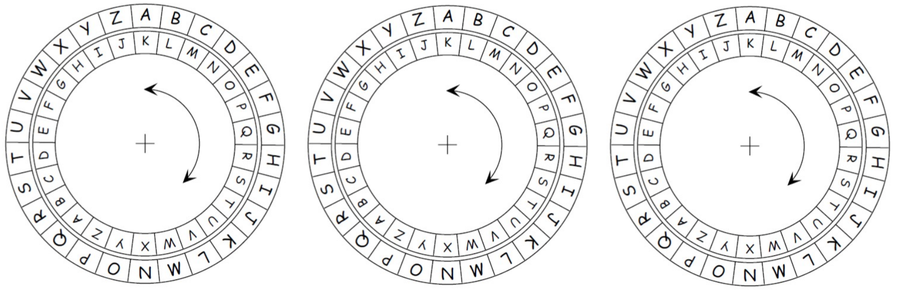
TD – Cryptanalyse de César

Source : http://www.etaletaculture.fr/



L	A	C	R	Y	P	T	O	L	O	G	I	E	E	S	T	L	A	S	C	I	E	N	C	E	D
U	S	E	C	R	E	T	E	L	L	E	E	N	G	L	O	B	E	L	A	C	R	Y	P	T	O
G	R	A	P	H	I	E	L	E	C	R	I	T	U	R	E	S	E	C	R	E	T	E	E	T	L
A	C	R	Y	P	T	A	N	A	L	Y	S	E	L	A	N	A	L	Y	S	E	D	E	C	E	T
T	E	D	E	R	N	I	E	R	E																

Système de Vigenère

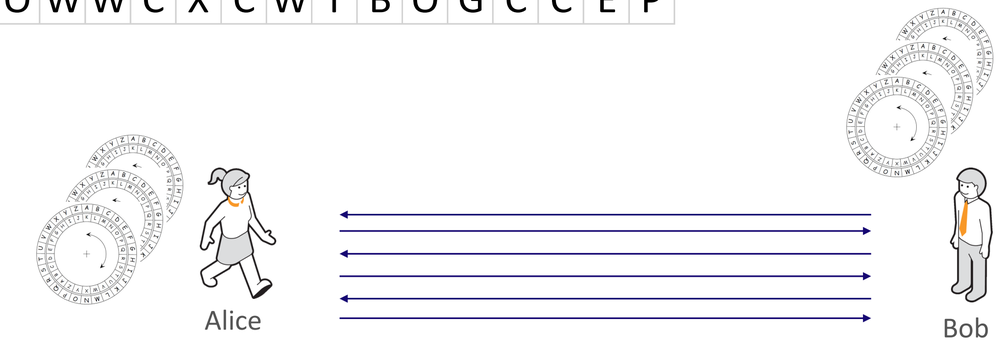


- Proposé au XVIème siècle par Blaise de Vigenère
- Principe : utiliser successivement plusieurs systèmes de César (plusieurs substitutions) avec des décalages différents
- Exemple avec la clé KEY ⇒ A = K, A = E, A = Y, A = K, A = E, A = Y, …
- Exemple de texte clair

C	E	M	E	S	S	A	G	E	E	S	T	C	H	I	F	F	R	E	A	V	E	C	L	E	S	Y	S	T	E	M	E	D	E	C	E	S	A	R

- Et du chiffré correspondant

M	I	K	O	W	Q	K	K	C	O	W	R	M	L	G	P	J	P	O	E	T	O	G	J	O	W	W	C	X	C	W	I	B	O	G	C	C	E	P



TD – Cryptanalyse de Vigenère

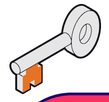
- Texte chiffré avec Vigenère :

X	Z	M	E	T	Y	P	P	F	G	W	P	G	N	M	V	L	X
K	Z	R	U	F	V	N	P	X	J	P	Q	G	O	I	U	D	G
K	P	R	E	P	W	S	F	M	U	P	V	C	T	X	C	D	W
G	K	X	G	X	I	T	L	M	T	P	T	Q	F	V	C	Q	J
K	C	Q	G	C	U	W	P	R	Q	F	W	E	Z	R	P	L	T
U	D	S	P	D	I	V	A	I	T	N	I	X	Z	R	U	E	S
W	E	I	U	W	I	U	Q	S	T	N	I	U	E	S	W	E	I
U	W	I	U	Z	R	F	P	W	G	E	X	Q	F	W	N	P	W
O	Z	C	G	Y	W	F	P	G	Q	X	Q	W	Y	M	E	L	X
K	Z	R	U	S	Y	D	P	V	V	C	I	G	G	I	U	L	W
V	C	S	R	S	C	U	T	G	K	P	R						

Réponse TD – Cryptanalyse de Vigenère

- Cryptanalyse indépendamment proposée par Babbage (1854) et Kasiki (1863)
- Une même lettre est toujours chiffrée de la même façon sur chaque substitution

1. Trouver le nombre de substitutions utilisées
   - Chercher des répétitions de séquences (par exemple de 3 lettres), en faisant le pari qu'elle correspond à un même texte clair chiffré avec la même clé
   - Regarder la distance entre ces répétitions en faisant le pari que c'est un multiple du nombre de substitutions
   - Prendre le diviseur commun et avec une bonne probabilité, c'est le nombre de substitutions
2. Utiliser la cryptanalyse de César
   - En faire autant qu'il y a de substitutions, en utilisant le résultat de l'étape précédente



Technique de cryptanalyse toujours utilisée aujourd'hui

Réponse TD – Cryptanalyse de Vigenère

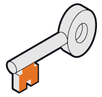
X	Z	M	E	T	Y	P	P	F	G	W	P	G	N	M	V	L	X
K	Z	R	U	F	V	N	P	X	J	P	Q	G	O	I	U	D	G
K	P	R	E	P	W	S	F	M	U	P	V	C	T	X	C	D	W
G	K	X	G	X	I	T	L	M	T	P	T	Q	F	V	C	Q	J
K	C	Q	G	C	U	W	P	R	Q	F	W	E	Z	R	P	L	T
U	D	S	P	D	I	V	A	I	T	N	I	X	Z	R	U	E	S
W	E	I	U	W	I	U	Q	S	T	N	I	U	E	S	W	E	I
U	W	I	U	Z	R	F	P	W	G	E	X	Q	F	W	N	P	W
O	Z	C	G	Y	W	F	P	G	Q	X	Q	W	Y	M	E	L	X
K	Z	R	U	S	Y	D	P	V	V	C	I	G	G	I	U	L	W
V	C	S	R	S	C	U	T	G	K	P	R						

Séquence	Position	Distance	Décomposition
EIU	110-125	15	5.3
TNI	100-118	18	3.3.2
GKP	36-189	153	17.3.3
UES	106-121	15	5.3
KZR	19-163	144	3.3.2.2.2.2
QFW	82-139	144	19.3

- PGCD = 3
- Taille probable de la clé : 3
- Cryptanalyse fréquentielle possible sur les 3 tranches

Système de Vernam

- Proposé en 1917 par Gilbert Vernam
- Même principe que Vigenère, dans lequel le nombre de substitutions est la longueur du message
- Ce n'est cependant pas suffisant !
  - Il faut que chaque substitution soit parfaitement aléatoire
  - Il faut que chaque ensemble de substitutions ne soit utilisé qu'une seule fois



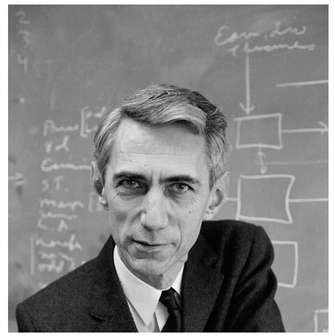
Notion de masque jetable : One-Time Pad OTP

Système de Vernam

$\oplus$	0	1
0	0	1
1	1	0

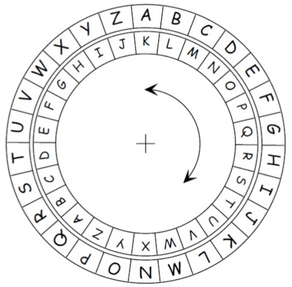
- En binaire, utilisation de l'opération de « ou exclusif » : $\oplus$
- Soit M un message à chiffrer et soit K un élément de la taille de M (appelée la clé)
- Le chiffré est alors $\mathrm{C} = \mathrm{M} \oplus \mathrm{K}$
- Pour retrouver M connaissant C et K, il suffit de calculer $\mathrm{M} = \mathrm{C} \oplus \mathrm{K}$ $(= M \oplus K \oplus \mathrm{K} = \mathrm{M})$

Sécurité inconditionnelle du OTP



- Formulée et prouvée par Claude Shannon en 1949
- Notion de sécurité inconditionnelle
  - La connaissance du chiffré ne fournit aucune information sur le clair
  - $Pr[M = m_0 | C = c_0] = Pr[M = m_0]$ pour tout message m_0 et tout chiffré c_0
- Utilisations réelles… (?)
  - Espions russes pendant la guerre froide avec des carnets rouges pas très aléatoires
  - Téléphone rouge entre Américains et Russes pendant la guerre froide

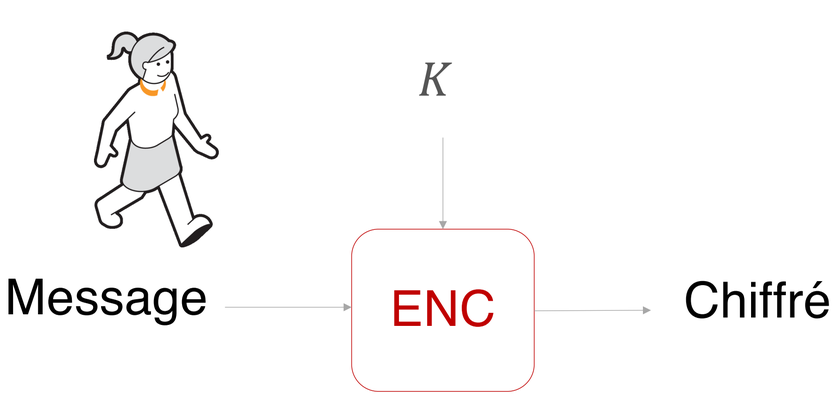
19ème et début 20ème – Systèmes composés

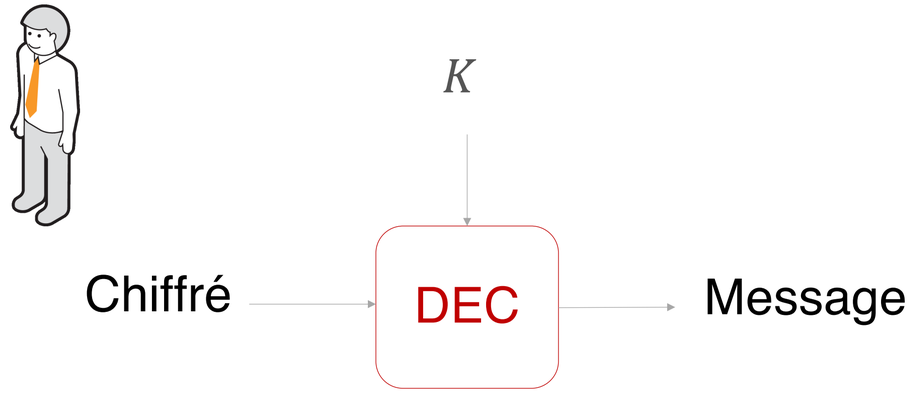


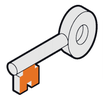
- Utilisation majeure de systèmes composés
- Objectif : augmenter la difficulté de la cryptanalyse en utilisant plusieurs opérations secrètes de chiffrement en série
- Exemples : double substitution, double transposition, substitution suivie par une transposition, etc.
- Démarrage aussi de l'âge technique
  - Objectif : les systèmes devenant plus complexes, il faut simplifier l'utilisation pour les humains
  - Exemple de la machine Enigma utilisée par les nazis

Concept de cryptographie à clé secrète

- Transformation d'un message en clair M en un message chiffré C en utilisant une clé secrète K
- La fonction de chiffrement doit être inversible
- La même clé est utilisée pour chiffrer et pour déchiffrer



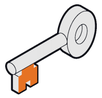




C'est le concept de cryptographie à clé secrète (ou cryptographie symétrique)

Principes fondamentaux de la cryptographie à clé secrète moderne

- Proposés par Claude Shannon, et largement utilisés aujourd'hui
- La confusion vise à cacher n'importe quelle structure algébrique dans le système
  - Assurée par une substitution non-linéaire (on parle souvent de boîte-S)
  - Il doit s'agir de la seule étape non-linéaire, afin de gagner en efficacité
- La diffusion permet à chaque bit du message d'influer une grande partie du texte chiffré
  - La modification d'un bit en entrée modifie de nombreux bits en sortie
  - Assurée par une permutation linéaire

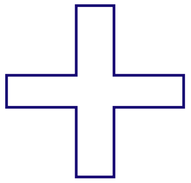


Toujours utilisé aujourd'hui dans le design de systèmes cryptographiques

Construction générique

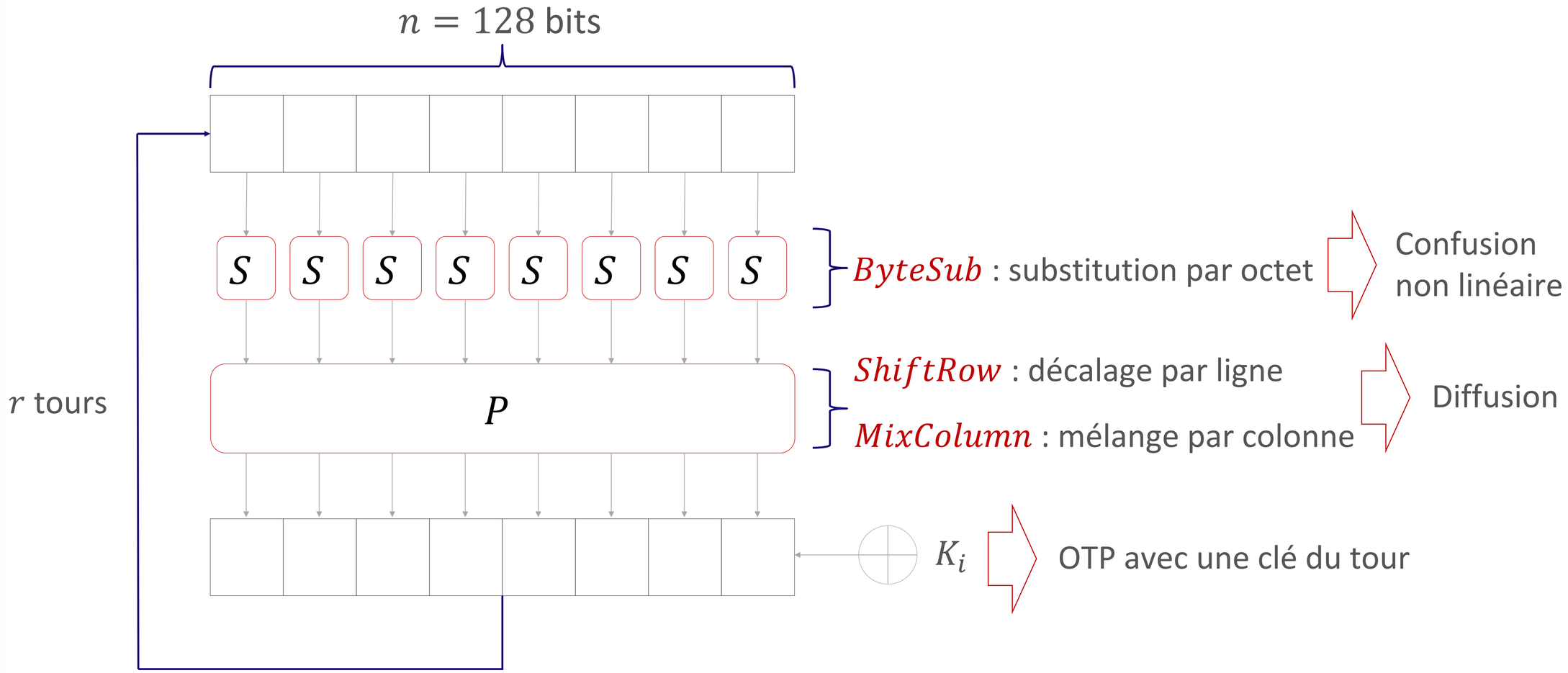
Construction : combiner substitutions et permutations sur plusieurs tours

1. OTP avec une clé dépendante du tour : $M \oplus K_i$
2. Découper le résultat de n bits en plusieurs sous-chaînes de longueur l et exécuter la substitution non-linéaire $\pi_S: \{0,1\}^l \rightarrow \{0,1\}^l$
3. Recoller les sous-chaînes et appliquer la permutation $\pi_P: \{1, \cdots, n\} \rightarrow \{1, , n\}$
4. Recommencer avec le message résultant, et une nouvelle clé de tour



Déchiffrement : opérations inverses, en utilisant les mêmes clés de tour

Illustration avec l'algorithme AES

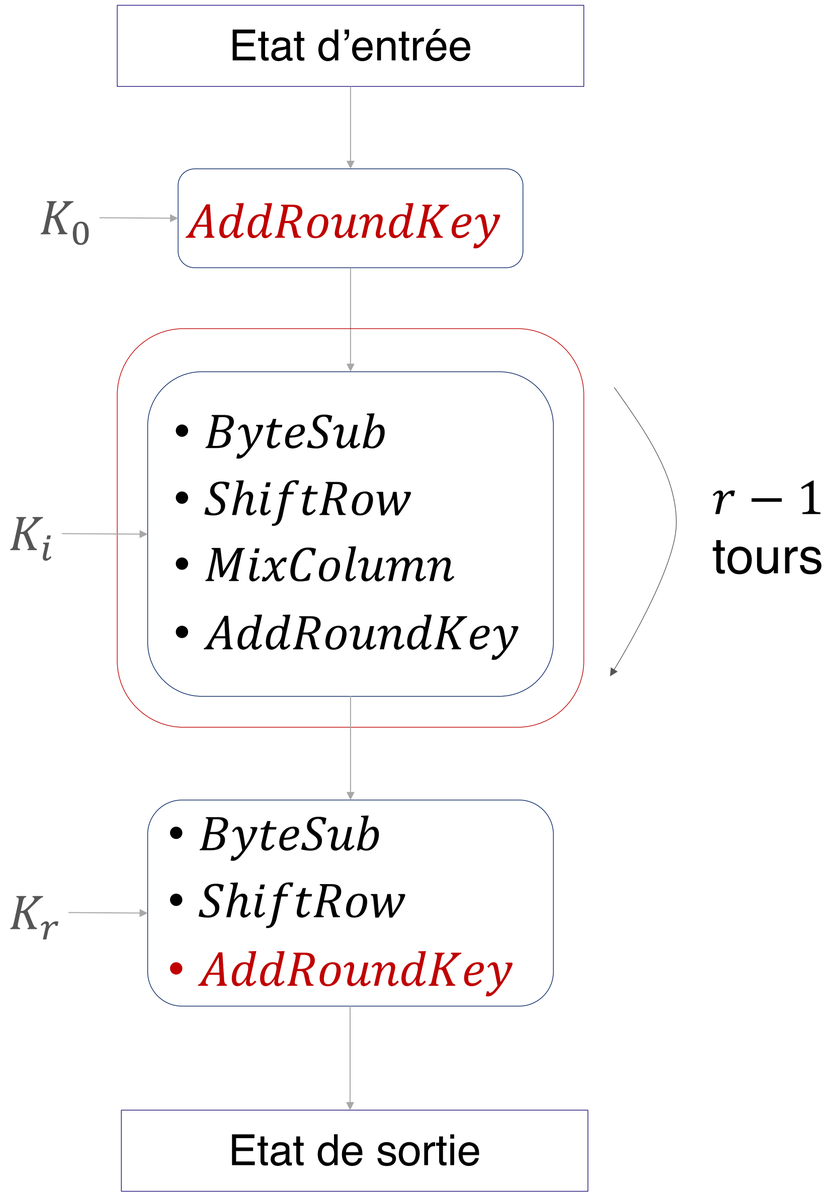


	$k = 128$	$k = 256$
$n = 128$	$r = 10$	$r = 14$

Quelques détails sur AES

État AES :

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$



Fonction $ByteSub$

- Fonction non-linéaire fixe de 8 bits vers 8 bits
  - Exécutée sur chaque octet de l'état, avec la même fonction
- Basée sur une opération algébrique dans $GF(2^8) = \mathbb{Z}_2[X]/P$ avec $P(X) = X^8 + X^4 + X^3 + X + 1$
  - Chaque octet va être représenté par un élément de $GF(2^8)$ (réduits modulo P, de degré < 8)
  - Exemple : $a = 01010100 \rightarrow X^6 + X^4 + X^2$ (Attention : les coefficients sont dans $\mathbb{Z}_2 = \{0,1\}$!)
- Construction de $ByteSub$
  - Calculer l'inverse de l'octet a dans $GF(2^8) = F_2[X]/P$
  - Calculer l'image du résultat précédent par la fonction $G: GF(2^8) \rightarrow GF(2^8)$ telle que

$$G(X) = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot X + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix} \text{ où } X = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}$$

Utilisation d'une table de correspondance

- En pratique, il est souvent plus simple d'utiliser une table de correspondance pour la fonction $ByteSub$

poids faible

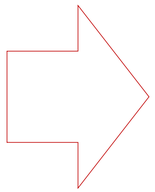
poids fort

	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	3^E	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

Fonction $ShiftRow$

- Cette transformation effectue un décalage cyclique des lignes de l'état selon différents offsets
  - La ligne 0 n'est pas décalée
  - La ligne 1 est décalée de 1 octets
  - La ligne 2 est décalée de 2 octets
  - La ligne 3 est décalée de 3 octets
- Il s'agit d'une simple permutation $\Rightarrow$ diffusion au niveau des octets

$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$



$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
$a_{1,1}$	$a_{1,2}$	$a_{1,3}$	$a_{1,0}$
$a_{2,2}$	$a_{2,3}$	$a_{2,0}$	$a_{2,1}$
$a_{3,3}$	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$

Fonction $MixColumn$

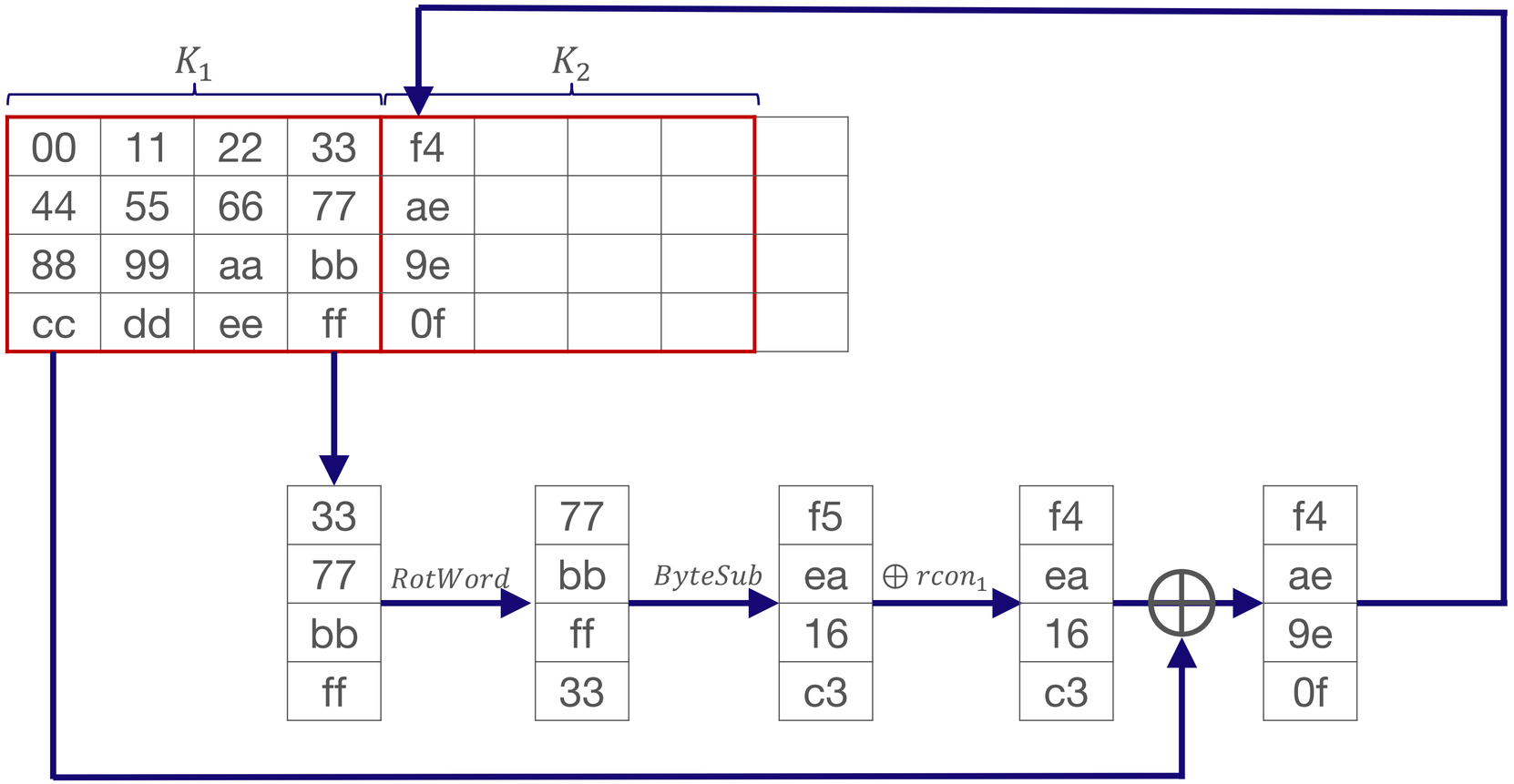
- Cette transformation consiste à prendre chaque colonne de l'état et à exécuter la fonction $H: \left(GF(2^8)\right)^4 \rightarrow \left(GF(2^8)\right)^4$ telle que

$$H(X) = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 00 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot X$$

- Nous devons à nouveau utiliser la multiplication dans $GF(2^8)$ avec $P(X) = X^8 + X^4 + X^3 + X + 1$
- Cette étape est linéaire et participe aussi à la diffusion (cette fois-ci des bits)

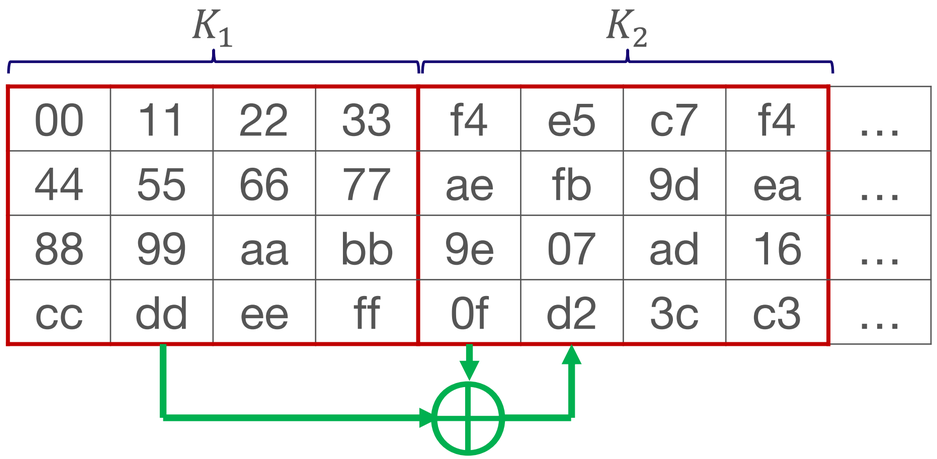
$AddRoundKey$ et calcul de la clé étendue

- La fonction $AddRoundKey$ sur chaque octet de l'état : $\forall(i,j), a_{i,j} = a_{i,j} \oplus k_{i,j}$
- Exemple de calcul de la clé étendue : $K \rightarrow \{K_i\}$



$AddRoundKey$ et calcul de la clé étendue

- La fonction $AddRoundKey$ sur chaque octet de l'état : $\forall(i,j), a_{i,j} = a_{i,j} \oplus k_{i,j}$
- Exemple de calcul de la clé étendue : $K \rightarrow \{K_i\}$



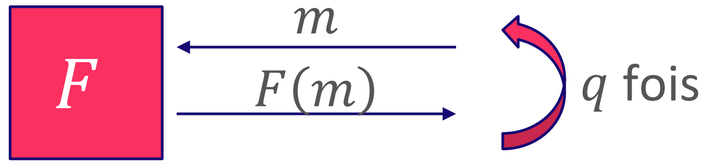
Que nous dit la théorie sur sa sécurité…

Notion de chiffrement idéal

Challenger

Attaquant

- $b \in_R \{0,1\}$
- If $b = 0$, $F = E_k$
- If $b = 1$, $F =$ permutation aléatoire sur $\{0,1\}^n$



$\tilde{b}$

- $\tilde{b} = b$?

- L'outil idéal pour protéger une information ⇒ Confidentialité des données
- Mais difficile à prouver formellement…

Sécurité par l’absence d’attaque

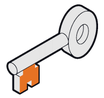
- Un algorithme secret permet-il moins d’attaques… ?

Notion de clé – Principes de Kerckhoffs

- Sortir de la cryptographie par l'obscurité : principes de Kerckhoffs
- Enoncé de six desiderata de la « cryptographie militaire » en 1883 !
  - Trois portent sur la facilité d'usage, un sur l'algorithme de chiffrement
  - Deux portent sur la notion de clé

« La sécurité d'un système ne doit pas être fondée sur son caractère secret »

« Seule une donnée de petite taille (clé) doit assurer la sécurité »



On préfère un système dont l'algorithme est publié qu'un système dont l'algorithme est secret

- La sécurité de la clé devient essentielle
  - Ne pas utiliser la clé trop longtemps, mais elle est moins coûteuse à changer

Sécurité par l'absence d'attaque

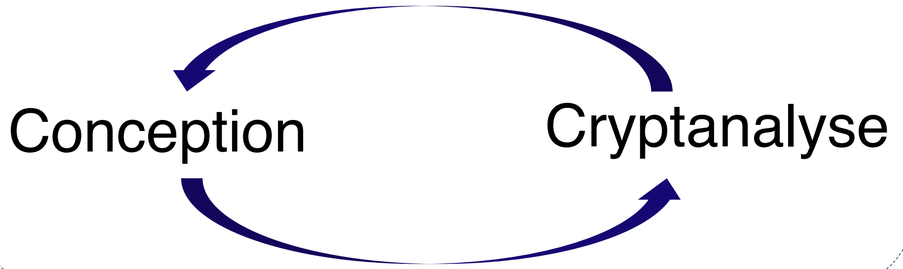
Objectifs

- ~~Un algorithme secret permet-il moins d'attaques… ?~~ NON
- Attaques ciblées contre un algorithme particulier
- Attaques génériques (ne dépendent pas de l'algorithme)
  - Principes de Kerckhoffs ⇒ attaque par force brute
- Techniques d'attaques classiques : différentielle, linéaire

Moyens

- Obtenir des couples (entrée, sortie) pour une même clé
- Plus il a de couples, plus fort il sera

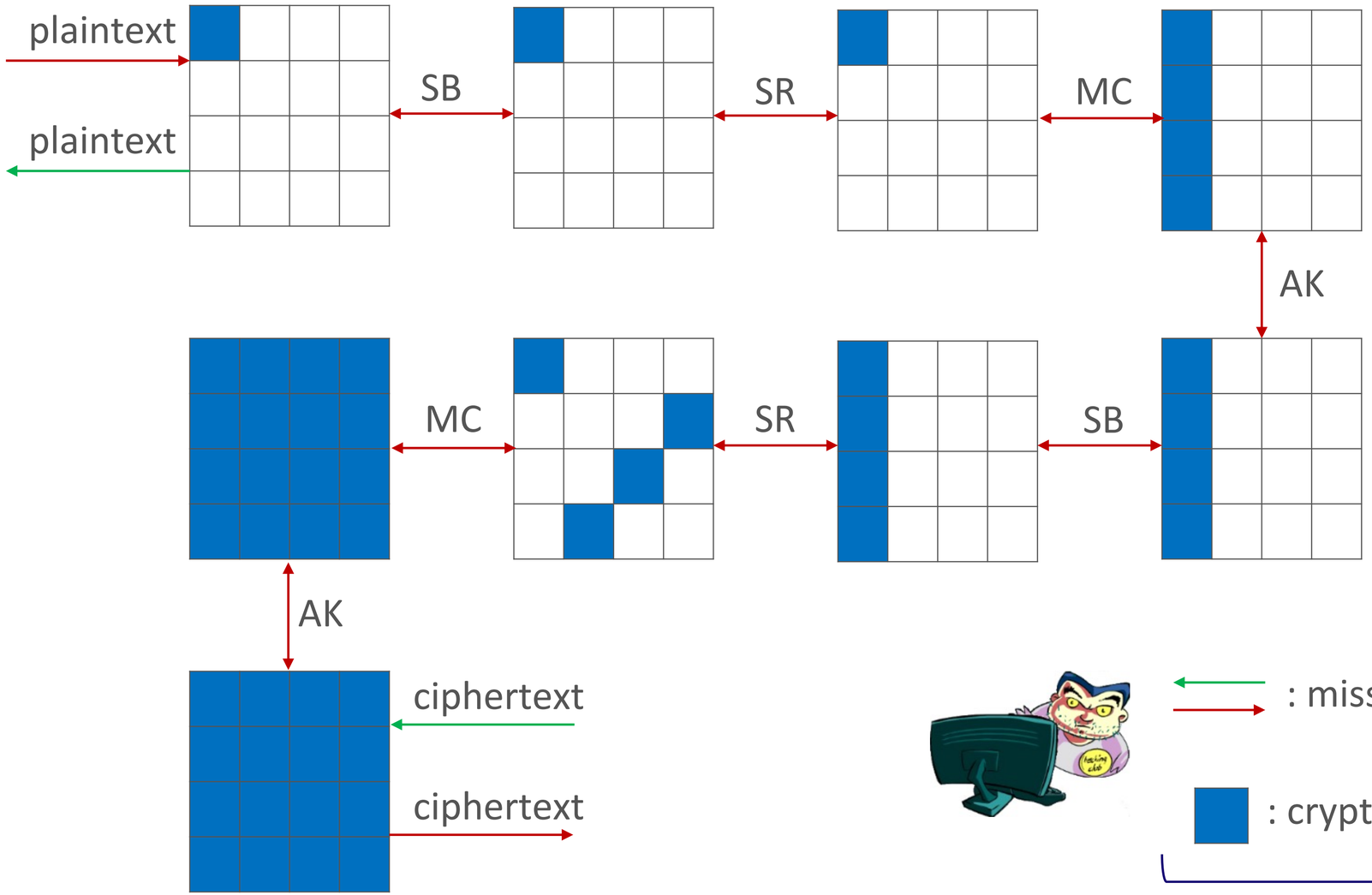
Cryptographie à clé secrète



Attaque par force brute

- Objectif : tester toutes les clés jusqu'à tomber sur la bonne
  - Hypothèse : je connais un couple (M, C) et je cherche K tel que $C = ENC(K, M)$
- Exemple avec AES
  - Un ordinateur Intel Core 2 Extreme QX9770, 3.2 GHz effectue 60000 MIPS
  - Une exécution d' AES nécessite environ 1200 instructions élémentaires
  - Pour tester une clé, il faut $\frac{1200}{6 \cdot 10^{10}} \approx 20 \cdot 10^{-9} = 20$ ns
  - Pour tester les 2^{128} clés, il faut $20 \cdot 10^{-9} \cdot 2^{128} \approx 2^{102}$ sec $\approx 10^{23}$ années !
- On estime aujourd'hui que l'on peut effectuer 2^{80} opérations, mais pas 2^{128}

Technique de cryptanalyse contre AES



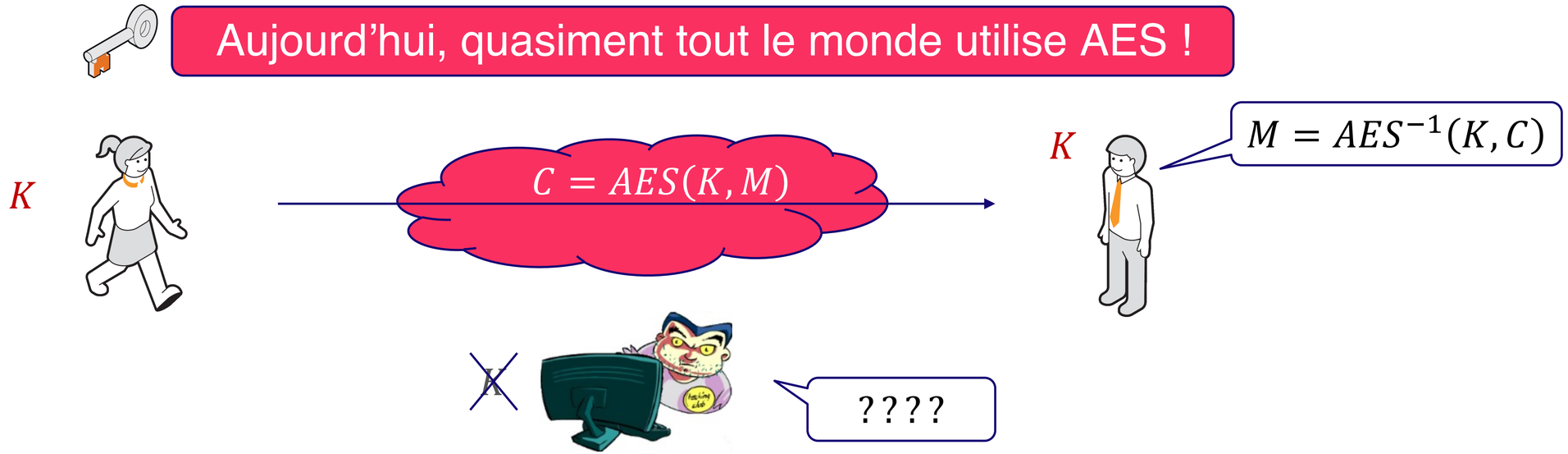
: miss/meet in the middle

: cryptanalyse différentielle

$$T[i][j] = |\{x: f(x) \oplus f(x \oplus i) = j\}|$$

AES est-il l'algorithme magique ?

- AES est né d'une compétition organisée par le NIST à la fin des années 90
- Standards de l'algorithme : FIPS PUB 197, ISO/IEC 18033-3
- Aucune attaque pertinente n'existe aujourd'hui sur cet algorithme

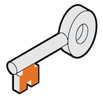


- Mais est-il suffisant pour Alice et Bob ?

2. Comment traiter des messages plus gros, et faire plus que la confidentialité ?

Chiffrement par blocs et agencement

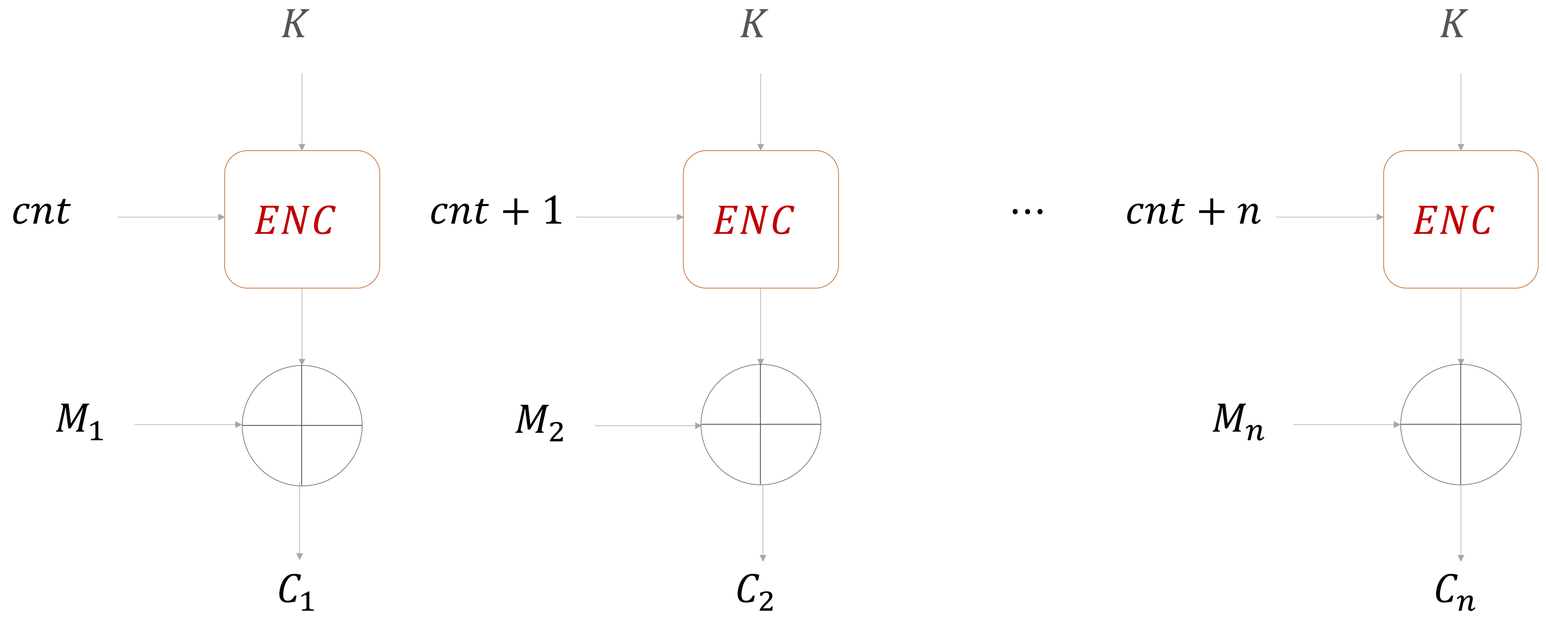
- Les systèmes par blocs permettent de traiter des petits blocs
- L'agencement des blocs va permettre des tailles quelconques de messages, et d'assurer
  - La confidentialité : c'est la propriété de base des systèmes par blocs, mais l'agencement des blocs va permettre des approches plus sûres
  - L'authenticité : est-ce que le message provient bien de son détenteur ?
  - L'intégrité : est-ce que le message n'a pas été modifié ?



Un système cryptographique à clé secrète n'est rien sans ces modes opératoires

Mode opératoire et chiffrement par blocs: CTR

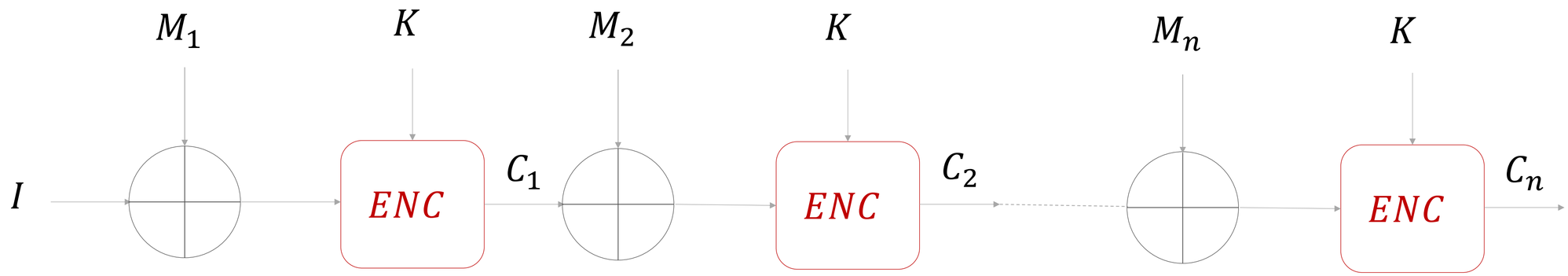
- Message : $M_1||M_2||\cdots||M_n$ où chaque M_n est un bloc de taille t



- Pré-calculs possibles, système parallélisable
- Contrôle d'intégrité impossible avec ce système : C_1 peut être remplacé par $C_1 \oplus \widetilde{M}$ pour envoyer le message $M_1 \oplus \widetilde{M}$

Mode CBC (Cipher Block Chaining)

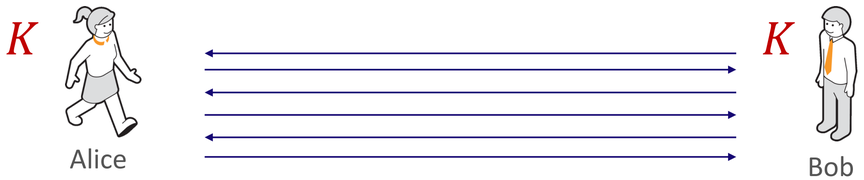
- Message : $M_1||M_2||\cdots||M_n$ où chaque M_n est un bloc de taille t
- Chiffrement : $\forall i, C_i = ENC(M_i \oplus C_{i-1}, K)$ avec $C_0 = I$ un vecteur d'initialisation
- Déchiffrement : $\forall i, M_i = DEC(C_i, K) \oplus C_{i-1}$



Mode $CBC\text{-}MAC$

- Le mode CBC n'est pas préconisé pour la confidentialité
- Mais il donne un MAC (*Message Authentication Code*) : Mode $CBC\text{-}MAC$
  - Alice calcule $\sum = C_n = CBC\text{-}MAC(M, K)$ et envoie $(\sum, M)$ à Bob
  - Bob calcule $CBC\text{-}MAC(M, K)$ et vérifie qu'il obtient la valeur $\sum$
- Que nous apporte ce mode ?
  - Bob sait que le message n'a pas été modifié ⇒ intégrité
  - Bob sait qu'il provient bien d'Alice, puisque ce n'est pas lui qui l'a généré, c'est donc elle ⇒ authenticité
  - Mais il n'y a a priori aucune confidentialité sur le message
- Standards: ANSI X9.9, ANSI X9.19, ISO 8731-1, ISO/IEC 9797-1

Excellentes performances



- Nous savons gérer des messages de taille quelconque
  - $AES128\text{-}CTR$ pour assurer la confidentialité
  - $AES128\text{-}CBCMAC$ pour l'authenticité et l'intégrité

	Intel Xeon 3,10 GHz, 64-bit	ARM Cortex M0+, 48 MHz
$AES128\text{-}CBC$ (table)	135 Mo/s	198 Ko/s
$AES128\text{-}CBC$	32 Mo/s	75 Ko/s
$AES128\text{-}CTR$ (table)	127 Mo/s	194 Ko/s
$AES128\text{-}CTR$	32 Mo/s	75 Ko/s

- Mais pour que ces systèmes cryptographiques fonctionnent, n'avons-nous pas oublié un petit détail concernant la clé K ?

3. Comment Alice et Bob peuvent-ils partager une même clé ?

Problématique de l'échange de clé

- L'échange de clé doit assurer la confidentialité et l'intégrité de la clé échangée
- Mais ce n'est pas suffisant : il faut partager la clé avec le bon interlocuteur
  - Nécessité d'une authentification qui va différer d'une technique à une autre
  - Notion d'accord de clé authentifié (*Authenticated Key Agreement AKA)*
  - Prise en compte du rejeu d'une authentification

1. Utiliser une tierce partie de confiance

- L'échange de clé passe par un « Centre de Distribution de Clé »
- Exemple de Kerberos

2. Utiliser une clé pré-partagée

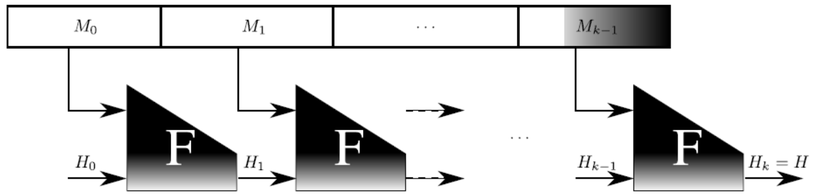
- Clé partagée dans un environnement sécurisé
- Il reste à voir comment utiliser cette clé

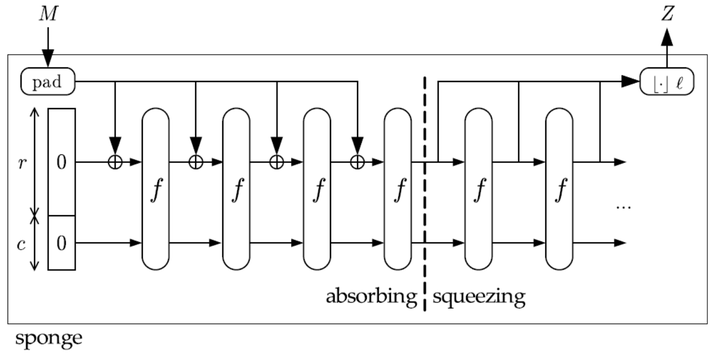
3...

Clé pré-partagée et usure de clé

- L'utilisation de la même clé pendant une longue période augmente les risques de compromission
  - Beaucoup de sorties peut servir à casser l'algorithme cryptographique
- Deux principes importants
  - Il faut donc changer la clé régulièrement
  - Une clé ne doit servir qu'à un unique besoin
- Dans le contexte d'une clé pré-partagée, ce n'est pas toujours facile de changer la configuration initiale, ni de stocker plusieurs clés
  - Notion de diversification d'une clé

Les fonctions de hachage





sponge

- Définition initiale
  - Fonction $H: \{0,1\}^* \rightarrow \{0,1\}^k$ pour k fixe
  - Connaissant M, le calcul de $H(M)$ doit être facile
- Fonction de hachage cryptographiquement sûre
  - Résistance à la pré-image
    - Infaisable de calculer M connaissant $H(M)$
  - Résistance à la seconde pré-image
    - Connaissant M et $H(M)$, infaisable de trouver M'tel que $H(M) = H(M')$ et $M \neq M'$
  - Résistance aux collisions
    - Infaisable de trouver M et M' tels que $H(M) = H(M')$

Détails de la fonction $HMAC$

- Il y a des similitudes entre une fonction de hachage et un MAC
- Pour une clé K et un message M
- Clé $K' = \begin{cases} H(K)\ si\ K\ est\ supérieur\ au\ bloc \\ K\ sinon \end{cases}$
- Fonction $HMAC(M,K) = H\left((K' \oplus opad)||H\left((K' \oplus ipad)||M\right)\right)$ avec H une fonction de hachage
- Mêmes propriétés que le $CBC\text{-}MAC$
- Standards : FIPS PUB 198, RFC 2104, RFC 2202

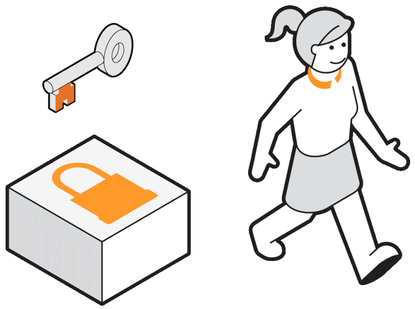
De multiples applications aux fonctions de hachage

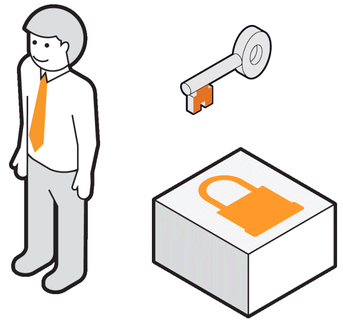
- Vérification de l'intégrité d'un message
- Code de détection de modification (pour des logiciels téléchargés)
- Fonction de dérivation d'une clé (*Hash-based Key Derivation Function HKDF*)
  - A partir d'une autre clé : $K_S = HKDF(K, contexte)$
  - En utilisant par exemple la fonction HMAC
  - Standards : NIST SP800-56Cr2, RFC 5869

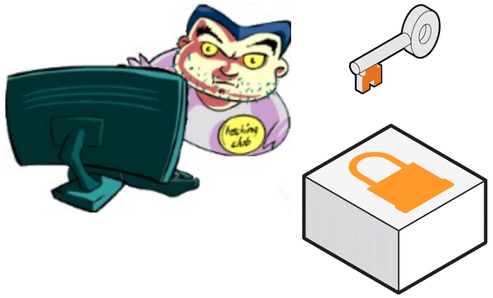
4. Et si une clé pré-partagée ou un tiers ne sont pas possible ?

TD – Concept de cryptographie à clé publique

- Faisons un jeu…

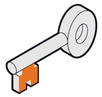






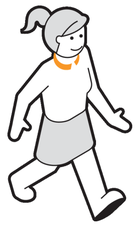
Concept de cryptographie à clé publique

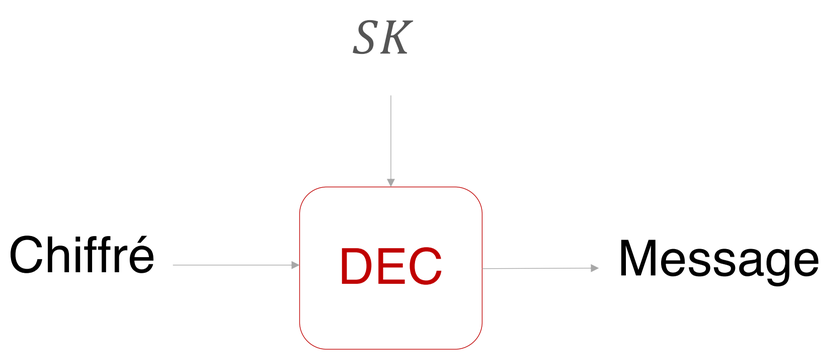
- L'étape de chiffrement n'implique a priori aucune connaissance particulière
- Diffie et Hellman (1976) : « Rendre la clé de chiffrement publique »



C'est le concept de cryptographie à clé publique (ou cryptographie asymétrique)

- Une clé est publique PK peut être utilisée par n'importe qui $\Longrightarrow$ étape non sensible
- Une clé reste secrète SK (ou privée) $\Longrightarrow$ étape sensible





- Le fait qu'une unique entité connaisse la clé privée, et que l'autre clé devienne publique, rend possible le concept de signature numérique

Concept de chiffrement

Chiffrement

- Génération des clés KeyGen
  - Entrée : paramètre de sécurité λ
  - Sorties : clé de chiffrement ek et clé de déchiffrement dk
- Chiffrement Enc
  - Entrées : message m et clé de chiffrement ek
  - Sortie : chiffré c
- Déchiffrement Dec
  - Entrées : chiffré c et clé de déchiffrement dk
  - Sortie : message m

- Probabiliste vs. Déterministe
  - Probabiliste si le chiffrement prend en entrée un aléa r
  - Cet aléa n'est pas forcément nécessaire lors de la phase de déchiffrement
- Sécurité
  - Chiffrement à sens unique
  - Chiffrement indistingable

Concept de signature

Signature

- Génération des clés KeyGen
  - Entrée : paramètre de sécurité λ
  - Sorties : clé de signature sk et clé de vérification vk
- Signature $Sign$
  - Entrées : message m et clé de signature sk
  - Sortie : signature σ
- Vérification $Verif$
  - Entrées : message m, signature σ et clé de vérification vk
  - Sortie : bit $0/1$

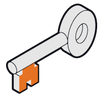
- Assure authentification de la source, intégrité et non-répudiation
- Equivalent juridiquement à une signature manuscrite
- Uniquement possible en cryptographie à clé publique : $sk \neq vk$
- Toujours probabiliste
- Sécurité
  - Non falsification

Principe du Certificat

- Une autorité signe le fait que la clé publique appartient bien à Alice

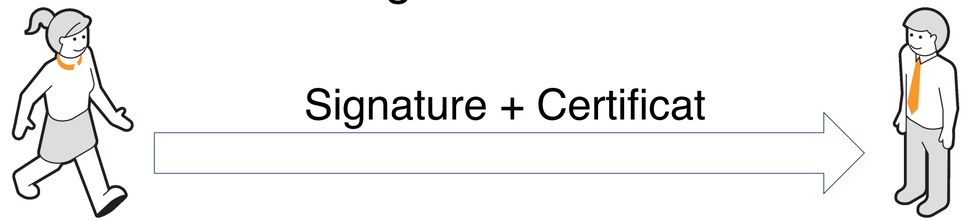
La clé publique d'Alice est pk_A

Certified
By Authority



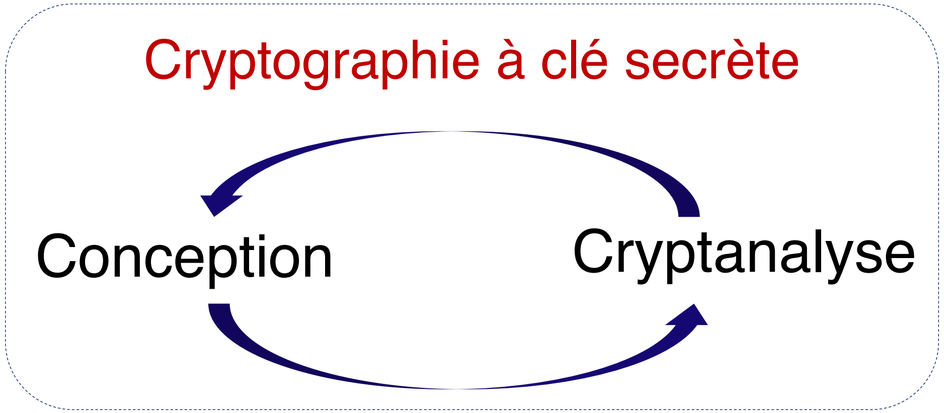
Notion d'infrastructure à clé publique (*Public Key Infrastructure PKI*)

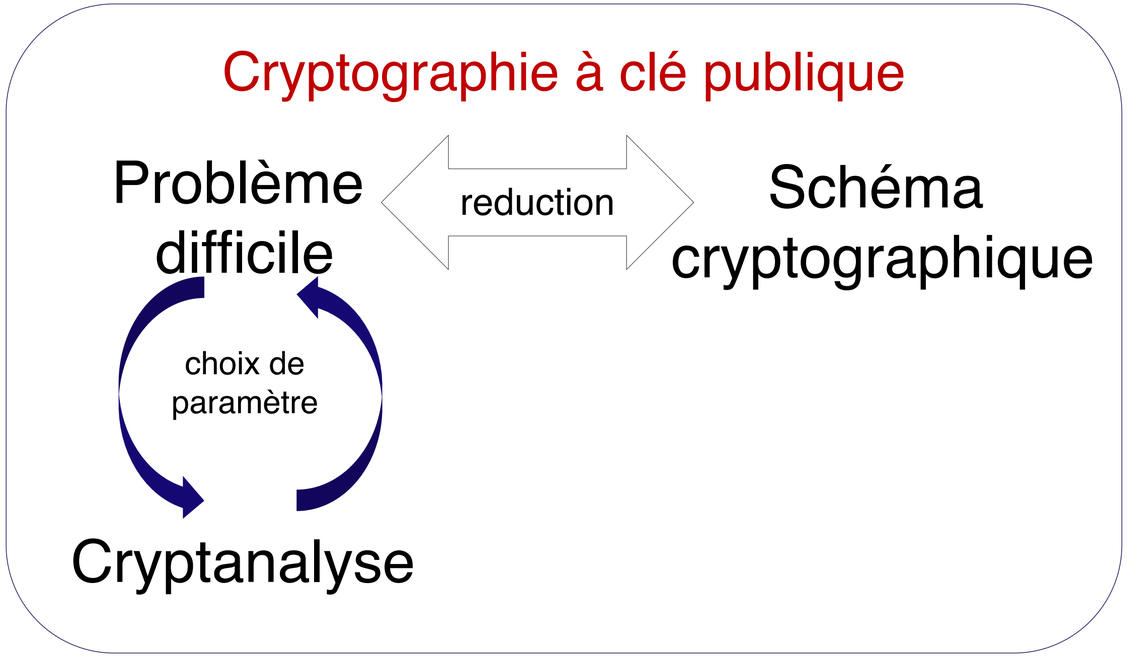
- Signature numérique
  - La signature est calculée avec la clé privée
  - La clé publique associée est certifiée par une autorité
  - Bob doit vérifier la validité du certificat, puis la validité de la signature



- Chiffrement
  - Le certificat contient la clé publique et est certifiée par une autorité. Il doit être vérifié par Bob avant usage
  - Le chiffrement est calculé avec la clé publique
  - Alice déchiffre en utilisant la clé privée associée

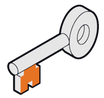
Deux designs conceptuellement différents





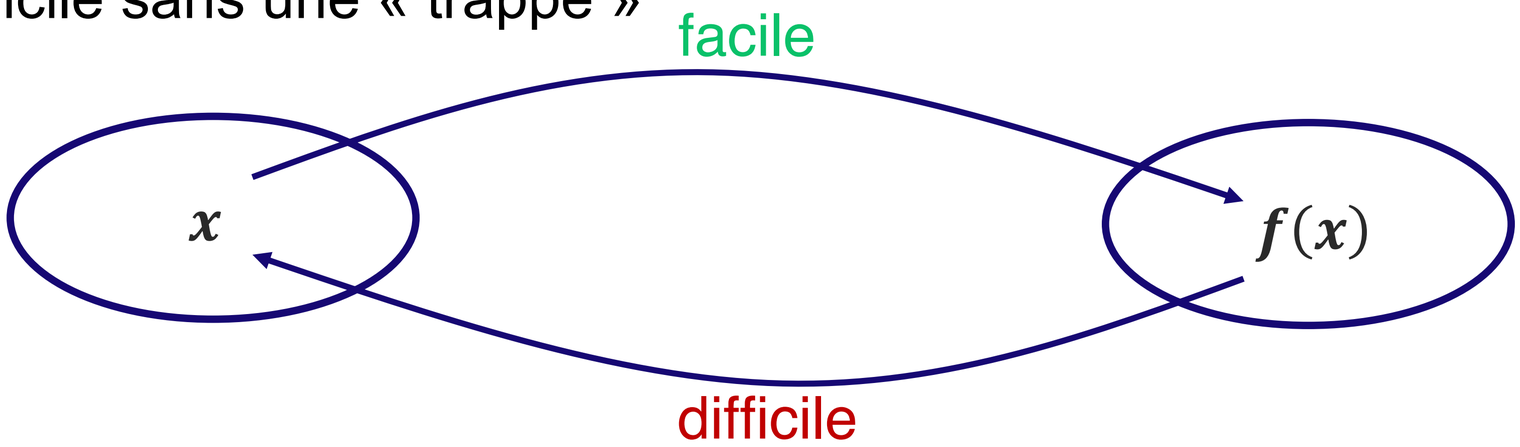
Attardons-nous un peu sur la notion de problème difficile…

Fonctions à sens unique



Fondement des systèmes cryptographiques à clé publique

- Principe
  - Pour une relation $y=f(x)$, calculer y est facile, et retrouver x à partir de y est difficile sans une « trappe »



- Comment construire de telles fonctions ?
  - A l'aide des mathématiques…

Calcul d'un logarithme discret

- Problème
  - Soit $(\mathbb{G},\cdot)$ un groupe d'ordre p
  - Alors $(\mathbb{G}, p, g, x) \rightarrow y = g^x$ est facile ⇒ exponentiation
  - Et $(\mathbb{G}, p, g, y) \rightarrow x \in \mathbb{Z}_p$ t.q. $y = g^x$ difficile ⇒ logarithme discret
- Cryptographie basée sur le logarithme discret

Usage autorisé	Jusqu'en 2021	Après 2021
Si la taille minimale d'une clé privée asymétrique dans un groupe est	**200 bits**	**200 bits**
Si la taille minimale du module premier du groupe est	**2048 bits**	**3072 bits**

- Recherche exhaustive ⇒ $\mathcal{O}(N)$
- Baby-Step-Giant-Step ⇒ $\mathcal{O}\left(\sqrt{N}\right)$
- Pollard's rho ⇒ $\mathcal{O}\left(\sqrt{N}\right)$

TD – Système de chiffrement ElGamal

- Standard : ISO/IEC 18033-6
- Génération des clés
  - Choisir un groupe $(\mathbb{G},\cdot)$ d'ordre p
  - Choisir la clé secrète $sk = x \in \mathbb{Z}_p$
  - Choisir la clé publique $pk = y = g^x$
- Chiffrement d'un message $m \in \mathbb{G}$
  - Commençons par un One-Time-Pad…

Instances possibles

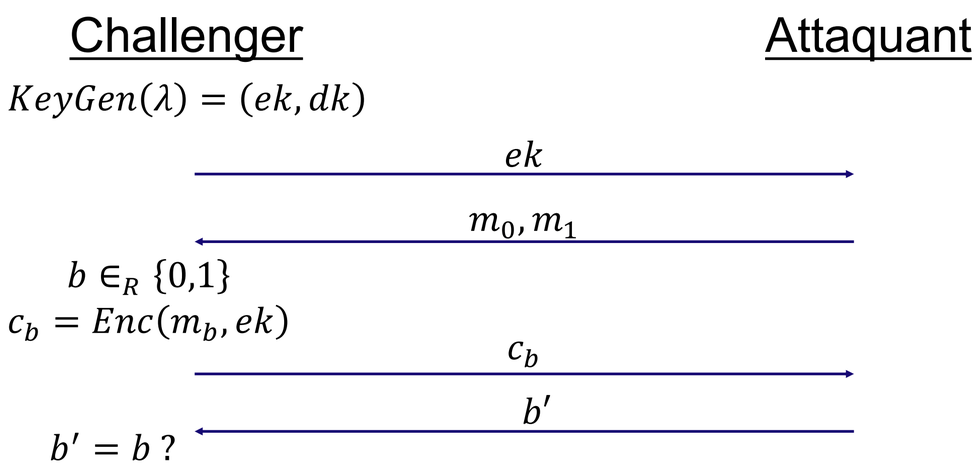
- Groupe multiplicatif $\mathbb{Z}_p^*$ des entiers modulo un nombre premier p
- Groupe des éléments inversibles $\mathbb{Z}_N^*$ pour un entier N composé
- Groupe des points entiers sur une courbe elliptique

TD – Système de chiffrement ElGamal

- Génération des clés
  - Choisir un groupe $(\mathbb{G},\cdot)$ d'ordre p
  - Choisir la clé secrète $sk = x \in \mathbb{Z}_p$
  - Choisir la clé publique $pk = y = g^x$
- Chiffrement d'un message $m \in \mathbb{G}$
  - Choisir un aléa $r \in \mathbb{Z}_p$
  - Calculer $T_1 = m \cdot y^r$ et $T_2 = g^r$
- Déchiffrement du chiffré (T_1, T_2)
  - Calculer $m = T_1 / T_2^x$

Notion de sécurité pour le chiffrement

Notion d'indistingabilité



- Moyens de l'attaquant
  - Texte clair choisi ($m \rightarrow c$) : notion d'IND-CPA
  - Texte chiffré choisi ($c \rightarrow m$) via un « oracle » : notion d'IND-CCA

Sécurité d'ElGamal

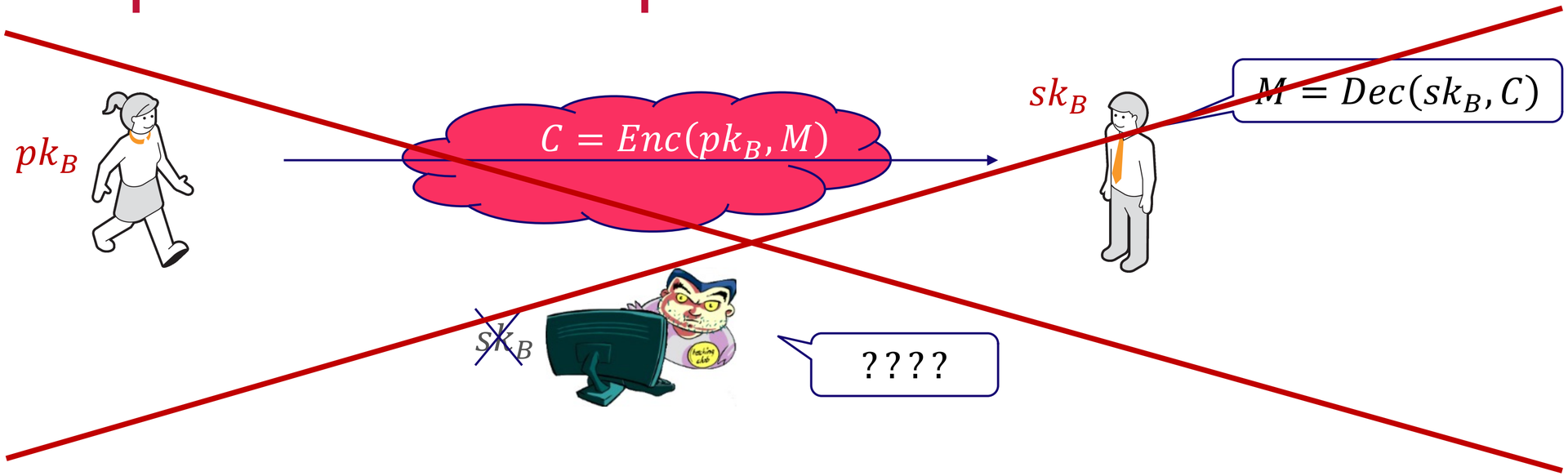
- Analyse
  - La recherche de la clé privée x à partir de la clé publique $y = g^x$ est équivalente au problème du logarithme discret
  - Si le problème du logarithme discret est résolu efficacement, alors ElGamal est cassé
  - Le contraire est peut-être faux car rien ne prouve qu'il ne peut pas être cassé par un autre moyen
- Réduction
  - Nous pouvons réduire l'indistingabilité d'ElGamal au problème DDH
  - Etant donnés $\mathbb{G}\ (\text{ordre}\ p), g, g^a, g^b, g^c\ (c \xleftarrow{\$} \mathbb{Z}_p)$ décider si $g^c = g^{\text{ab}}$

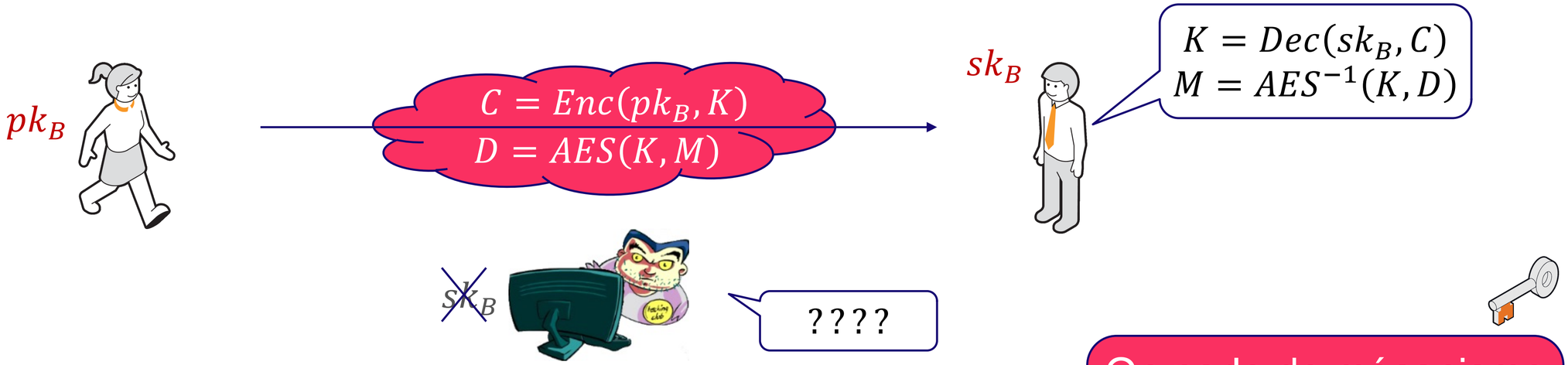
Performances d'ElGamal

- Performances sur un Intel P4 3 GHz (messages de 256 bits)
  - Chiffrement : 22 ms
  - Déchiffrement : 13 ms

	Intel P4 3 GHz, 32 bits	Intel Xeon 3,10 GHz, 64-bit	ARM Cortex M0+, 48 MHz
AES-128 CBC (table)		135 Mo/s	198 Ko/s
AES-128 CBC		32 Mo/s	75 Ko/s
AES-128 CTR (table)		127 Mo/s	194 Ko/s
AES-128 CTR		32 Mo/s	75 Ko/s
ElGamal Enc	1,4 Ko/s		
ElGamal Dec	2,4 Ko/s		

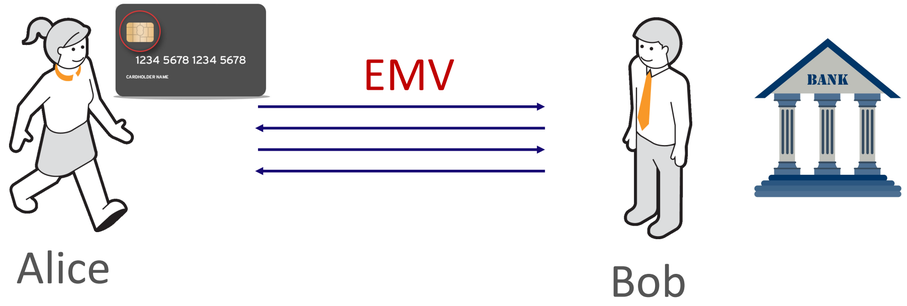
Conséquence de ces performances





On parle de mécanisme d'encapsulation de clé (KEM)

Schéma de signature de Schnorr



- Standard : ISO/IEC 14888-3
- Génération des clés
  - Choisir un groupe $(\mathbb{G},\cdot)$ d'ordre p
  - Choisir la clé secrète $sk = x \in \mathbb{Z}_N$
  - Choisir la clé publique $pk = y = g^x$
- Signature d'un message $m \in \mathbb{G}$
  - Choisir un aléa $r \in \mathbb{Z}_N$
  - Calculer $\mathrm{t} = g^r$, $c = \mathcal{H}(g \parallel y \parallel t \parallel m)$ et $\mathrm{s} = r - c \cdot x$
- Vérification de la signature (c, s) du message m
  - Vérifier que $c = \mathcal{H}(g \parallel y \parallel y^c \cdot g^s \parallel m)$

Notion de sécurité pour la signature

- Notion de non-falsification
  - Connaissant vk, un attaquant ne doit pas être capable de sortir un couple (m, σ) tel que

$$Verif(m, \sigma, vk) = 1$$

- Moyens de l'attaquant
  - Texte choisi $(m \rightarrow \sigma)$
- Qu'obtenons-nous ?
  - Authenticité de la provenance du message
  - Intégrité du message
  - Non-répudiation (nouveau par rapport à un MAC !)

Cryptographie à clé secrète vs. à clé publique

	Propriétés	Avantages	Inconvénients	Usages
Cryptographie à clé secrète	• Une clé secrète pour chaque paire d'entités • Même clé pour chiffrer et déchiffrer	• Rapide	• Beaucoup de clés à gérer • Pas de service de non-répudiation	• Chiffrement de grands volumes de données • Contrôle d'intégrité de grands volumes de données
Cryptographie à clé publique	• Une paire de clé (l'une publique et l'autre privée/secrète) pour chaque entité • Clé publique pour chiffrer, clé privée pour déchiffrer • Clé privée pour signer, clé publique pour vérifier	• Peu de clés à gérer • Service de non-répudiation fourni • Bonne structure mathématique	• Lent	• Distribution de clés secrètes • Signatures numériques • Cryptographie avancée

5. Implémentation et cryptographie

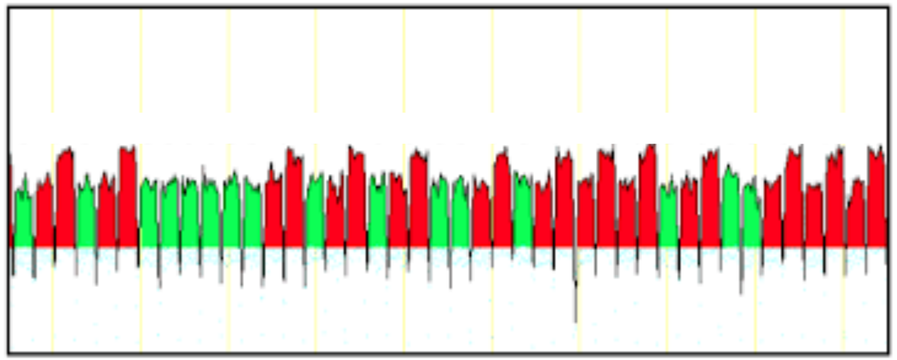
Revenons sur l'algorithme d'exponentiation x^n

- Idée : décomposer n en binaire $n = n_t \parallel n_{t-1} \parallel \cdots \parallel n_0$ où $n_t = 1$
- Algorithme
  - $R \leftarrow x$
  - for $(i = t - 1)$ to 0 do
    - $R \leftarrow R \times R$
    - if $(n_i == 1)$ then $R \leftarrow R \times x$
- TD : calculer 3^{11} avec cet algorithme

Attaques par canaux cachés

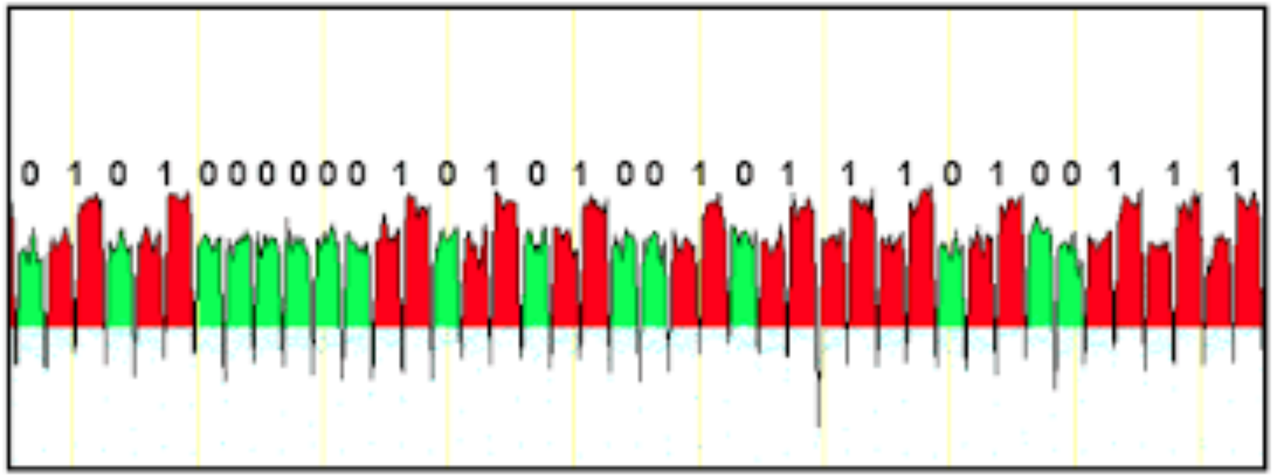
- $R \leftarrow x$
- for $(i = t - 1)$ to 0 do
  - $R \leftarrow R \times R$ → carré
  - if $(n_i == 1)$ then $R \leftarrow R \times x$ → multiplication

Analyse de la consommation électrique



Petit signal = carré
Grand signal = multiplication

Attaques par canaux cachés



- $R \leftarrow x$
- for $(i = t - 1)$ to 0 do
  - $R \leftarrow R \times R$
  - if $(n_i == 1)$ then $R \leftarrow R \times x$

- Vert = carré $\Rightarrow n_i = 0$
- Rouge = carré puis multiplication $\Rightarrow n_i = 1$
- On peut reconstituer la clé avec un simple oscilloscope
- Attaque qui fonctionne essentiellement avec les systèmes embarqués

Attaques par canaux cachés

- Idée : consommation constante
- Algorithme résistant à l'attaque précédente :
  - $R \leftarrow x$, $U \leftarrow R$
  - for $(i = t - 1)$ to 0 do
    - if $(n_i == 1)$ then $R \leftarrow R \times R$, $U \leftarrow R \times x$
    - if $(n_i == 1)$ then $R \leftarrow R \times R$, $R \leftarrow R \times x$
- Mais sujet à d'autres attaques par canaux cachés
  - Attaque par faute ⇒ utilisation de l'échelle de Montgomery
  - Attaque différentielle ⇒ nécessite des contremesures spécifiques

6. Cryptographie moderne

TD – Homomorphisme d’ElGamal

- Homomorphisme : une opération sur des chiffrées donne des opérations sur des clairs
- Montrer que le schéma de chiffrement d’ElGamal possède un certain homomorphisme
- Exploiter cette propriété pour montrer que ce schéma ne peut pas atteindre la propriété d’IND-CCA

TD – Homomorphisme d'ElGamal

- Etant donnés (T_1, T_2) le chiffré du message m et (D_1, D_2) le chiffré du message $\tilde{m}$, $(T_1 \cdot D_1, T_2 \cdot D_2)$ est le chiffré du message $m \cdot \tilde{m}$
  - En effet : $T_1 \cdot D_1 = (m \cdot \tilde{m}) \cdot y^{r+\tilde{r}}$ et $T_2 \cdot D_2 = g^{r+\tilde{r}}$
- Propriété d'IND-CCA : l'attaquant d'avoir accès à un oracle de déchiffrement
  - Calculer $D_1 = 2 \cdot y^{\tilde{r}}$ et $D_2 = g^{\tilde{r}}$
  - Envoyer $(T_1 \cdot D_1, T_2 \cdot D_2)$ à l'oracle de déchiffrement et récupérer $\tilde{m}$
  - Vérifier si $\tilde{m} = 2 \cdot m_0$ ou $\tilde{m} = 2 \cdot m_1$ et sortir le bit correspondant

- Mais cette propriété peut aussi être utilisée à bon escient…

TD – Chiffrement homomorphe et IA

- Prenons l'algorithme d'exponentiation vu tout à l'heure
- Nous considérons que le secret n est chiffré et nous savons faire des additions et des multiplications dans le monde chiffré
- Réécrire l'algorithme en notant les données à chiffrées

Algorithme A
Entrées : entiers x, n
Sortie : entier $R = x^n$
Exécution :
- $R \leftarrow x$;
- for $(i = t - 1$ to $0)$ do
  - $R \leftarrow R^2$;
  - If $(n_i == 1)$ then $R \leftarrow Rx$;
- output R;

Algorithme A
Entrées : entiers x, $[\![n]\!]$
Sortie : entier $R = x^n$
Exécution :

TD – Chiffrement homomorphe et IA

- Prenons l'algorithme d'exponentiation vu tout à l'heure
- Nous considérons que le secret n est chiffré et nous savons faire des additions et des multiplications dans le monde chiffré
- Maintenant, le réécrire en n'utilisant qu'additions et multiplications

Algorithme A
Entrées : entiers x, n
Sortie : entier $R = x^n$
Exécution :
- $R \leftarrow x$;
- for $(i = t - 1$ to $0)$ do
  - $R \leftarrow R^2$;
  - If $(n_i == 1)$ then $R \leftarrow Rx$;
- output R;

Algorithme A
Entrées : entiers x, $[\![n]\!]$
Sortie : entier $[\![R]\!] = [\![x^n]\!]$
Exécution :
- $[\![R]\!] \leftarrow x$;
- for $(i = t - 1$ to $0)$ do
  - $[\![R]\!] \leftarrow [\![R^2]\!]$;
  - If $([\![n_i]\!] == 1)$ then $[\![R]\!] \leftarrow [\![Rx]\!]$;
- output $[\![R]\!]$;

Algorithme A
Entrées : entiers x, $[\![n]\!]$
Sortie : entier $[\![R]\!] = [\![x^n]\!]$
Exécution :

TD – Chiffrement homomorphe et IA

- Prenons l'algorithme d'exponentiation vu tout à l'heure
- Nous considérons que le secret n est chiffré et nous savons faire des additions et des multiplications dans le monde chiffré
- Imaginez si vous avez aussi des boucles « while »…
- Standard (version draft) : ISO/IEC WD 18033-8

Algorithme A
Entrées : entiers x, n
Sortie : entier $R = x^n$
Exécution :
- $R \leftarrow x$;
- for ($i = t - 1$ to 0) do
  - $R \leftarrow R^2$;
  - If ($n_i == 1$) then $R \leftarrow Rx$;
- output R;

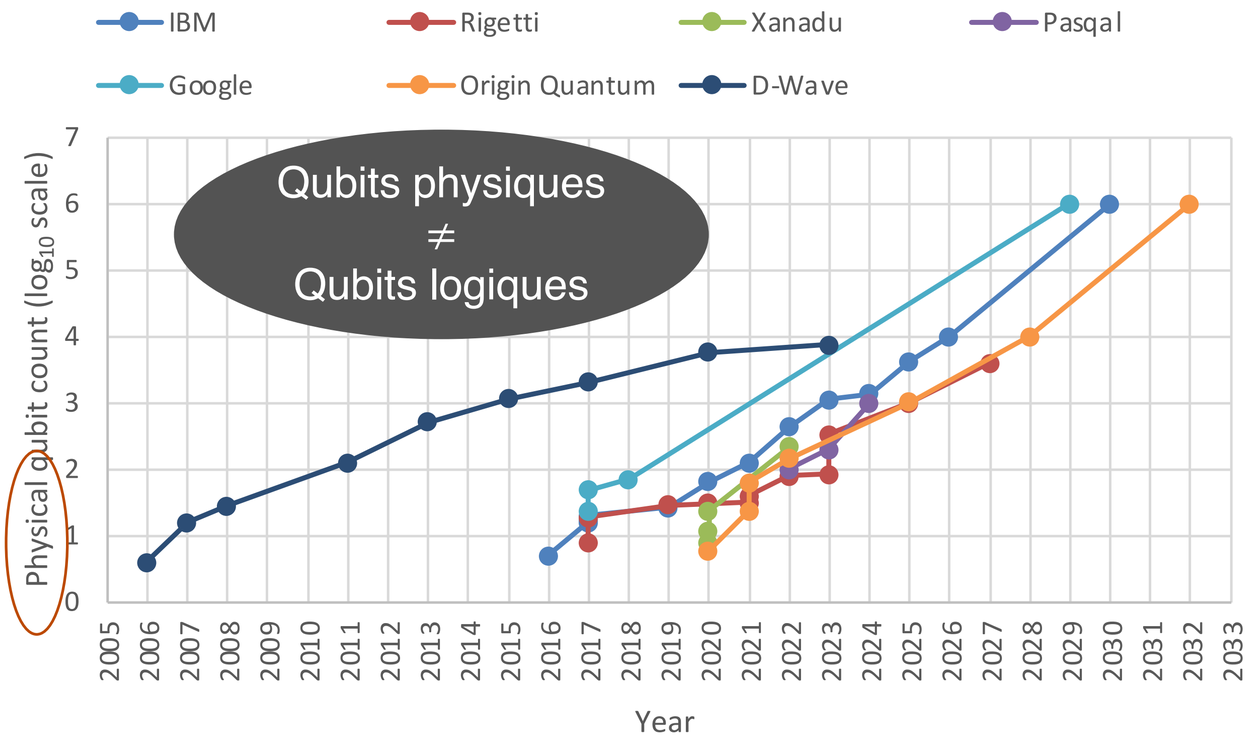
Algorithme A
Entrées : entiers x, $[\![n]\!]$
Sortie : entier $[\![R]\!] = [\![x^n]\!]$
Exécution :
- $[\![R]\!] \leftarrow x$;
- for ($i = t - 1$ to 0) do
  - $[\![R]\!] \leftarrow [\![R^2]\!]$;
  - If ($[\![n_i]\!] == 1$) then $[\![R]\!] \leftarrow [\![Rx]\!]$;
- output $[\![R]\!]$;

Algorithme A
Entrées : entiers x, $[\![n]\!]$
Sortie : entier $[\![R]\!] = [\![x^n]\!]$
Exécution :
- $R \leftarrow x$; $[\![R]\!] \leftarrow Enc(R)$;
- for ($i = t - 1$ to 0) do
  - $[\![R]\!] \leftarrow [\![R]\!] \cdot [\![R]\!]$;
  - $[\![R]\!] \leftarrow [\![R]\!] \cdot x \cdot [\![n_i]\!] + [\![R]\!] \cdot (1 - [\![n_i]\!])$;
- output $[\![R]\!]$;

Et les ordinateurs quantiques

Calcul quantique

- Un ordinateur quantique pense différemment
- Superposition et intrication quantiques
- Notion d'algorithme quantique : Grover, Shor, Simon, etc.



Cryptographie à clé publique

- Algorithme de Shor sur la factorisation et le logarithme discret ⇒ cassage total
- 6100 qubits logiques pour casser RSA 3072 bits
- 2300 qubits logiques pour casser ECC 256 bits

⇒ Pas une menace maintenant

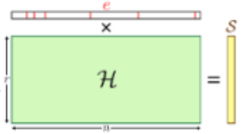
Paradigme du « stocker maintenant et déchiffrer plus tard »

Cryptographie post-quantique

- Cryptographie post-quantique créée à partir de problèmes mathématiques pour lesquels les ordinateurs quantiques ne sont pas plus puissants que les ordinateurs classiques

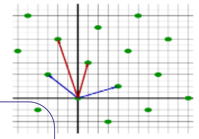
- Plusieurs solutions existent depuis plusieurs décennies

Codes

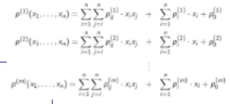


Arbres de hachage

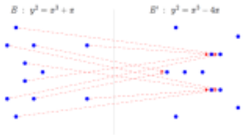
Réseaux euclidiens



Polynômes multivariés



Isogénies



Signature : SPHINCS+

Chiffrement : Kyber

Signature : Dilithium, Falcon

Futurs standards FIPS

Une cryptographie moderne basée sur les standards

- Le premier standard en cryptographie : algorithme DES (Data Encryption Standard) en 1970
- Principe général de la standardisation de la cryptographie aujourd'hui

Algorithmes cryptographiques

NIST

ISO International Organization for Standardization

IEEE

IEC INTERNATIONAL ELECTROTECHNICAL COMMISSION

Usages de ces algorithmes

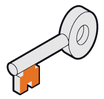
IETF

3GPP A GLOBAL INITIATIVE

ETSI

GSMA

- Une procédure aujourd'hui assez classique…
  - Compétition organisée par le NIST sur plusieurs années pour définir un standard
  - Adaptation, nouvelles propositions et usage par les autres (ISO/IEC, IEEE, …)
  - Il faut au moins 5 ans avant qu'un algorithme cryptographique ne soit standardisé



Toujours s'inspirer des standards et des recommandations des instances officielles (ANSSI, BSI, ENISA, NCSC, NSA, …)

Cryptographie et législation

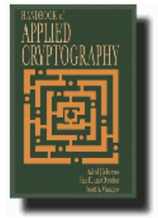
- Tout le monde ne peut pas utiliser la cryptologie totalement librement
- Il existe une réglementation en vigueur pour la commercialisation de produits intégrant des moyens de cryptologie
- Voir https://www.ssi.gouv.fr/entreprise/reglementation/controle-reglementaire-sur-la-cryptographie/

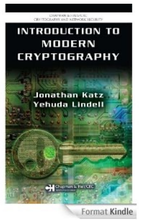
Conclusion

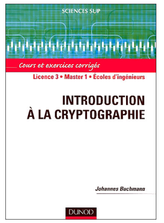
ATTENTION

- La cryptographie n’est qu’une composante de la sécurité
- Il en existe d’autres
  - Gestion des personnes, des organisations, des process
  - Gestion de l’implémentation matérielle (hardware)
  - Gestion de l’implémentation logicielle (software)

Références utiles







- Pour aller plus loin
  - Introduction to Cryptography, Johannes Buchmann, (2000)
  - Introduction to Modern Cryptography, Katz , Lindell (2007)
  - Handbook of Applied Cryptography, Alfred Menezes, Paul van Oorschot, Scott Vanstone (1996) http://cacr.uwaterloo.ca/hac/
- Référence des instances officielles
  - ANSSI en France : https://www.ssi.gouv.fr/guide/mecanismes-cryptographiques/
  - BSI en Allemagne : https://www.bsi.bund.de/EN/Themen/Unternehmen-und-Organisationen/Standards-und-Zertifizierung/Technische-Richtlinien/TR-nach-Thema-sortiert/tr02102/tr02102_node.html
  - ENISA en Europe : https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014
- Culture générale
  - L'histoire des codes secrets, Simon Singh, 1999

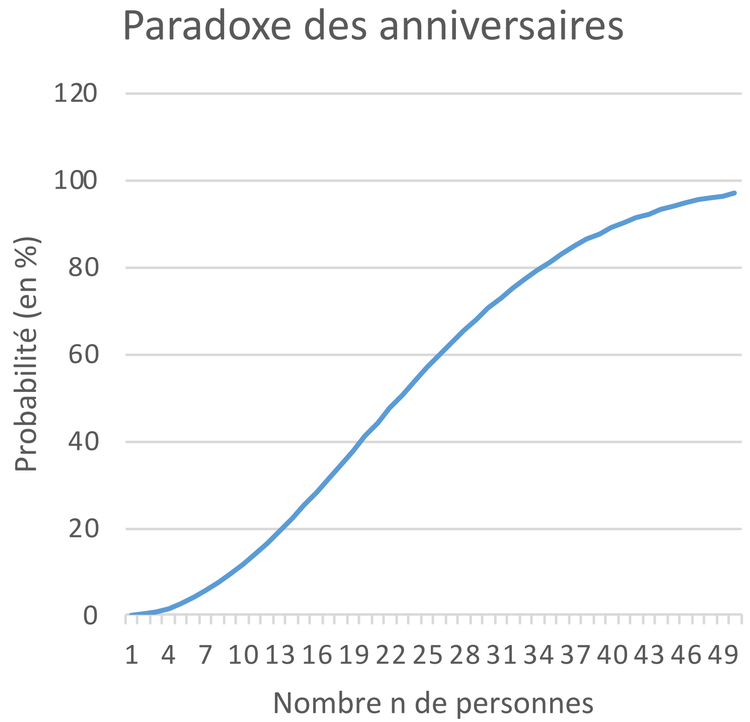
Merci

Résistance aux collisions et anniversaire

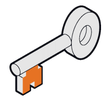
Idée générale

- Choisir M et M' au hasard et espérer que $H(M) = H(M')$
- Choisir deux personnes au hasard et espérer qu'elles aient la même date d'anniversaire
- Quelle probabilité ? Intuition = très faible

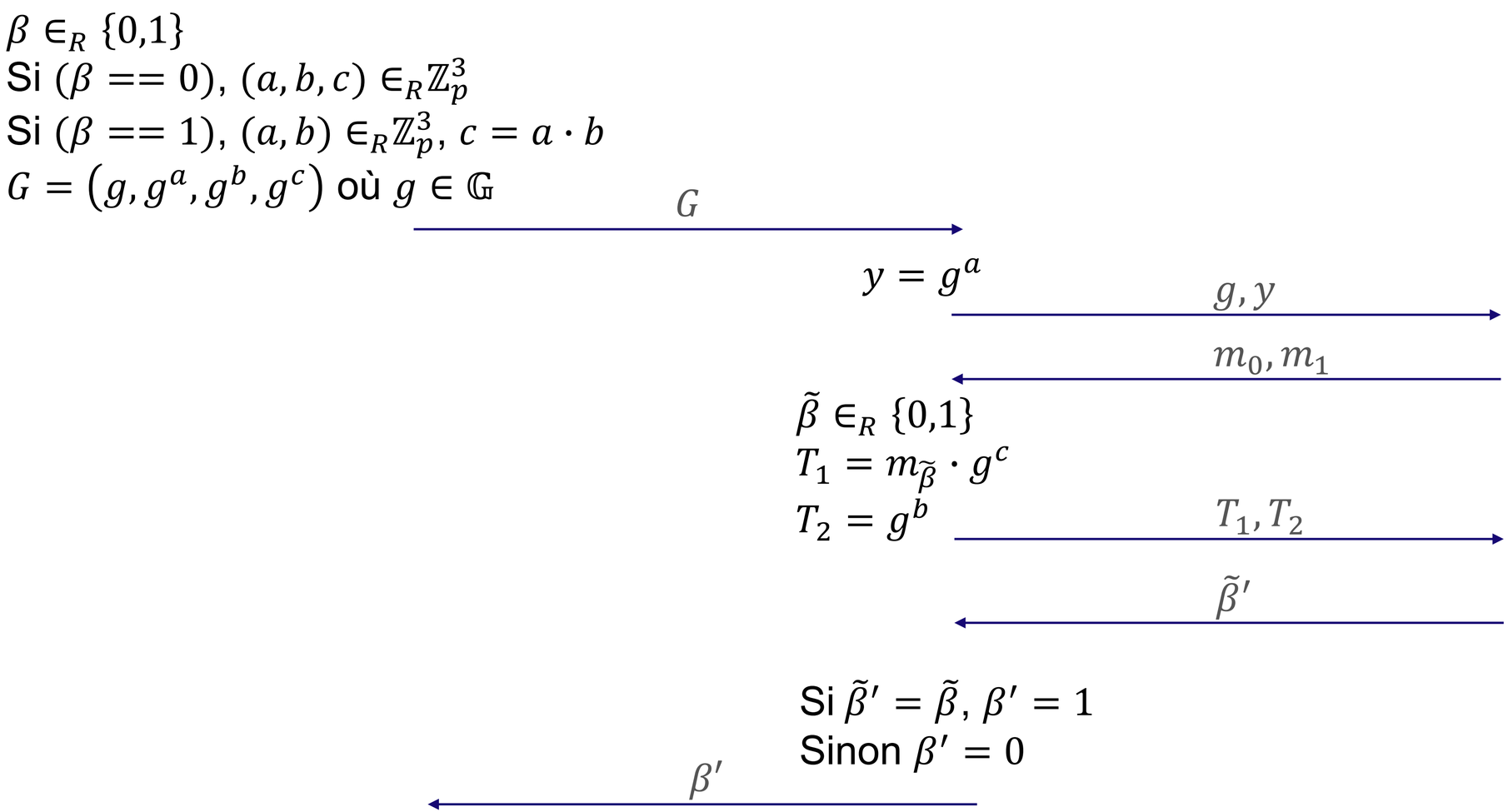
- Calculons cette probabilité $Pr(A)$ (pour 365 jours)
  - Probabilité $Pr(A')$ que dans un groupe de n personnes toutes aient un jour d'anniversaire différent
  - $Pr(A) = 1 - Pr(A')$
  - Evènement n : la personne n n'a pas la même date d'anniversaire que les personnes $\mathrm{n} - 1, \cdots, 2{,}1$
    - $Pr(1) = \frac{365}{365}$, $Pr(2) = \frac{364}{365}$, $Pr(3) = \frac{363}{365}$, $\cdots$, $Pr(n) = \frac{365-(n-1)}{365}$
  - Probabilités conditionnelles
    - $Pr(A') = \frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \times \cdots \times \frac{365-(n-1)}{365} = \frac{365!}{365^n(365-n)!}$
- C'est « paradoxal » !



Preuve de sécurité d'ElGamal



Technique de preuve très classique aujourd'hui



Preuve de sécurité d'ElGamal – Analyse

- Hypothèse : un attaquant contre ElGamal existe
  - Probabilité de réussite $= \frac{1}{2} + \varepsilon_{EG}$ avec ε_{EG} non négligeable
- Cas $c = ab$
  - Dans ce cas, c'est une vraie instance d'ElGamal : $Pr[\beta' = \beta] = \frac{1}{2} + \varepsilon_{EG}$
- Cas c aléatoire
  - Dans ce cas, ce n'est pas une instance d'ElGamal : $Pr[\beta' = \beta] = \frac{1}{2}$
- En utilisant les probabilités conditionnelles avec des évènements indépendants
  - Probabilité $Pr[\beta' = \beta] = \frac{1}{2}\left(\frac{1}{2} + \varepsilon_{EG}\right) + \frac{1}{2}\left(\frac{1}{2}\right) = \frac{1}{2} + \frac{\varepsilon_{EG}}{2}$ avec $\frac{\varepsilon_{EG}}{2}$ non négligeable