

# Cryptographie algorithmique

Didier Alquié  
[didier.alquie@laposte.net](mailto:didier.alquie@laposte.net)

Version de 07/2025

# Introduction, scénario, vocabulaire

# Se protéger

## Pourquoi ?

L'information est vulnérable

- À l'accidentel non intentionnel  
erreurs humaines, perturbations
- **À la malveillance intentionnelle**  
écoute, modification, altération, destruction, usurpation

## Comment ?

- Mesures organisationnelles ; coffre fort, sentinelle, etc.
- Mesures techniques :
  - sûreté de fonctionnement : contre l'accidentel
  - **SSI** :
    - sécurité informatique : détection d'intrusion, pare feu, anti-virus,
    - **cryptographie**

# Cryptologie

“*Étude de ce qui est caché*”. Techniques algorithmiques et mathématiques assurant des services de sécurité, principalement :

- ***Confidentialité*** : s'assurer que l'information n'a pas été divulguée.
- ***Intégrité*** : s'assurer que l'information n'a pas été modifiée.
- ***Authentification d'origine*** : s'assurer qu'une information provient bien d'où elle est censée provenir.
- ***Authentification d'entité*** : s'assurer qu'une entité est bien ce qu'elle prétend être.
- ***Non réputation*** : s'assurer qu'une entité ne peut pas se soustraire à un engagement.

# Cryptologie et al.

La cryptologie comprend :

- La *cryptographie*, qui a trait à la conception et à la construction des fonctions assurant les services de sécurité.  
D'où le nom d' "algorithmes cryptographiques"
- La *cryptanalyse* : qui vise à mettre en défaut les propriétés de sécurité en "cassant" les algorithmes cryptographiques. Elle peut
  - avoir une fin **opérationnelle**, contre un "ennemi"
  - être un outil **académique** d'évaluation afin de faire progresser l'état de l'art et d'améliorer par ricochet la qualité de la conception

*Que l'on soit "gentil" ou "espion", la cryptanalyse, en tant que science et technique, ne peut jamais être ignorée.*

## Scénario et acteurs traditionnels



- Alice et Bob sont des “gentils” qui veulent communiquer via un canal non sûr.
- Charlie est un “méchant” qui écoute ce qui passe sur le canal de transmission : *attaquant passif*.
- Eve est une “méchante” qui écoute et modifie ce qui passe sur le canal de transmission : *attaquant actif*.

# Les algorithmes cryptographiques

Ce sont des fonctions paramétrées par des clés secrètes.

- chiffrement-déchiffrement
- construction de motif d'intégrité
- signature-vérification (authentification, intégrité, non répudiation)

Ils s'appuient sur des briques élémentaires appelées *primitives*. Par exemple :

- objets mathématiques possédant des propriétés spécifiques
- ne rendent pas le service à elles seules . . .
- . . .mais en les assemblant correctement
- exemples : fonctions pseudo aléatoires, problèmes mathématiques réputés difficiles, etc.

## Spécification d'un algorithme

Description algébrique par des équations, toujours polynomiales binaires :

$$\text{chiffré } C = \text{Fonction}(\text{clé } K, \text{ clair } P)$$

Exemple jouet :  $K = (k_0, \dots, k_7)$ ,  $P = (p_0, \dots, p_7) \mapsto C = (c_0, \dots, c_7)$

$$\begin{aligned}c_0 &= \dots \oplus p_0 k_0 p_3 k_3 k_6 \oplus p_0 k_0 p_3 p_5 k_5 p_6 k_7 \oplus p_0 k_0 p_3 p_5 k_5 p_6 \oplus p_0 k_0 p_3 p_5 k_5 k_7 \oplus p_0 k_0 p_3 p_5 \dots \\c_1 &= \dots \oplus p_0 p_1 p_2 p_3 k_6 \oplus p_0 p_1 p_2 k_3 k_4 k_5 p_7 \oplus p_0 p_1 p_2 k_3 k_4 k_5 \oplus \dots \\c_2 &= \dots \oplus p_3 k_3 k_4 p_6 k_6 \oplus p_0 k_1 p_3 k_3 k_4 p_6 \oplus p_0 k_1 p_3 k_3 k_4 k_6 \oplus p_0 k_1 p_3 k_3 p_5 p_6 k_6 \oplus \dots \\c_3 &= \dots \oplus p_6 \oplus p_0 k_3 p_5 k_6 p_7 \oplus p_0 k_3 p_5 p_7 \oplus p_0 k_3 p_5 \oplus p_0 k_3 k_5 p_6 k_6 \oplus p_0 k_3 k_5 p_6 p_7 \oplus \dots \\c_4 &= \dots \oplus k_0 k_2 p_3 p_4 p_5 p_6 \oplus k_0 k_2 p_3 p_4 k_5 p_6 \oplus k_0 k_2 p_3 p_4 p_6 \oplus k_0 k_2 p_3 p_4 k_7 \oplus k_0 k_2 p_3 p_4 \dots \\c_5 &= \dots \oplus p_1 p_3 p_4 k_4 k_5 k_6 \oplus p_1 p_3 p_4 k_4 k_5 \oplus p_1 p_3 p_4 k_4 p_6 k_6 k_7 \oplus p_1 p_3 p_4 k_4 p_6 k_6 \oplus \dots \\c_6 &= \dots \oplus k_2 p_4 k_4 p_5 p_7 k_7 \oplus p_2 k_2 p_4 k_4 p_5 p_7 \oplus p_2 k_2 p_4 k_4 p_5 k_7 \oplus p_2 k_2 p_4 k_4 k_5 p_7 k_7 \dots \\c_7 &= \dots \oplus p_3 p_4 k_4 k_6 k_7 \oplus p_3 p_4 k_4 \oplus p_3 p_4 p_5 k_5 p_6 k_6 k_7 \oplus p_3 p_4 p_5 k_5 p_6 k_6 \oplus p_3 p_4 p_5 \dots\end{aligned}$$

# Symétrique/asymétrique

Cryptographie *symétrique*, ou *à clé secrète*.

- Alice et Bob partagent un secret commun préalable : la clé secrète , qui sert pour chiffrer et déchiffrer
- Alice et Bob jouent des rôles réversibles
- la cryptographie historique en fait partie

Cryptographie *asymétrique*, ou *à clé publique*.

- Née en 1976. Article de Diffie-Hellman
- la clé de chiffrement est différente de la clé de déchiffrement
- la clé de chiffrement est publique
- le clé de déchiffrement est secrète
- L'expéditeur et le destinataire jouent des rôles *a priori* non réversibles
- pas de secret commun préalable

## Plan du cours

# Plan du cours (1/5)

## Part. 1 - Techniques anciennes

- Substitutions alphabétiques
- Transposition

## Part. 2 - Attaquant, modèle de sécurité

- Approche intuitive
- Modèle d'attaquant
  - Capacités
  - Contexte
  - Défi
- Modèle de sécurité
  - Calculatoire
  - Conception vs évaluation

## Part. 3 - Chiffrement symétrique

- Chiffrement par flot
  - L'approche Shannon
  - Générateur de pseudo aléa
  - Registres linéaires et fonctions booléennes
  - Sécurité d'un GPA

# Plan du cours (2/5)

Chiffrement par bloc

    Primitive bloc

    Ingrédients d'une primitive bloc

    Sécurité d'une primitive bloc

    Mode d'opération

Sécurité d'un chiffrement symétrique

    Niveau de sécurité

    Rappel : sécurité sémantique

    Indistinguabilité

    Illustrations

Part. 4 - Chiffrement asymétrique

    Principe

    Sécurité d'un chiffrement asymétrique

        Niveau de sécurité

        Indistinguabilité

    Problèmes mathématiques et algorithmes

        Factorisation des entiers et RSA

        Logarithme discret et Elgamal

    Autour du chiffrement asymétrique

# Plan du cours (3/5)

Chiffrement hybride  
Négociation de clés Diffie-Hellman  
Certification de clé publique

## Part. 5 - Intégrité symétrique. Hachage

Intégrité symétrique. MAC  
Sécurité d'un MAC  
Fonctions de hachage  
Définition et sécurité  
Construction Merkle-Damgård  
Construction éponge

Constructions de MAC  
Avec fonctions de hachage  
Avec primitives bloc

Intégrité et confidentialité combinées  
Version de base  
Version AEAD

## Part. 6 - Signature

Principe

# Plan du cours (4/5)

Sécurité d'une signature

Exemples

Signature RSA et factorisation

Signature et logarithme discret

Part. 7 - Authentification

Mot de passe

Challenge/réponse

Zero knowledge

Part. 8 - Quantique et cryptographie

Ne pas confondre

Cryptographie quantique

Cryptographie post quantique

Impact sur la cryptographie symétrique

Impact sur la cryptographie asymétrique

Références

Annexes mathématiques et algorithmique

# Plan du cours (5/5)

## Mathématiques

Généralités

Deux résultats de probabilités

Algèbre générale

Polynômes

Arithmétique des entiers

Arithmétique modulaire

## Algorithmique

Tri, listes

Calcul algébrique

Factorisation, logarithme discret

## En un coup d'œil

	confidentialité	intégrité	authentification	non-réputation
chiffrement symétrique	✓	✗	✗	✗
chiffrement asymétrique	✓	✗	✗	✗
hachage	✗	✓	✗	✗
MAC	✗	✓	✗✓	✗
chiffrement symétrique intègre	✓	✓	✗✓	✗
signature	✗	✓	✓	✓

– Partie 1 –  
Chiffrements anciens

# Chiffrements anciens

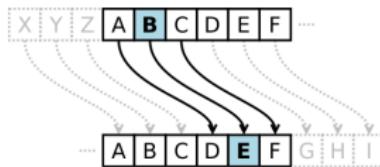
1 Substitutions alphabétiques

2 Transposition

## Substitution simple mono alphabétique

### Le chiffrement par décalage

- Permutation circulaire dont la clé est l'amplitude du décalage (+3 dans le chiffrement de César traditionnel) :



clé :      +3

clair :    CECI EST UN MESSAGE SECRET

chiffré : FHFL HVW XQ PHVVDJH VHFUHW

- Cryptanalyse par recherche exhaustive de la clé : complexité 26

## Chiffrement par permutation quelconque de l'alphabet

- la clé est la permutation tout entière

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

- Chiffrement

clair    CECI EST UN MESSAGE SECRET

chiffré    TBTR BFW NY HBFFLJB FBTOBW

- Cryptanalyse :

- Recherche exhaustive de la clé : complexité  $26! \approx 2^{88}$
- Attaque fréquentielle : basée sur la statistique d'apparition des lettres dans le texte clair.

## Variantes

### *Substitution homophonique*

- Pour une équirépartition des lettres du texte chiffré, les lettres les plus fréquentes sont chiffrées de plusieurs façons

### *Substitution par polygrammes*

- Le texte clair est chiffré par “paquets” de lettres
- la substitution individuelle n'a plus de sens.

Encore cryptanalysable à chiffré seul par les attaques fréquentielles.

## Substitution poly alphabétique : chiffrement de Vigenère

### Description

- La clé est un mot secret, répété autant de fois que nécessaire pour s'adapter à la longueur du texte clair.
- chiffrement caractère par caractère, par combinaison suivant la table de Vigenère

clair :

CECI EST UN MESSAGE SECRET

mot clé "TRACE" répété :

TRAC ETR AC ETRACET RACETR

chiffré :

VVCK XJT YG MGWLRRGG LVCTIM

### Description moderne

- Les lettres sont codées par les entiers modulo 26  
 $a=0, b=1, \dots, z=25$ .
- la combinaison suivant la table correspond à l'addition mod 26

# Table de Vigenère

Vigenère Cipher Table

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

Using the Table

Encryption	Decryption
Message: SEN <u>D</u> HELP	Ciphertext: T <u>Y</u> U <u>J</u> FFA
Key: BUL <u>GEB</u> UL	Key: BUL <u>GEB</u> UL
Ciphertext: T <u>Y</u> U <u>J</u> FFA	Message: SEN <u>D</u> HELP

Cryptanalyse : recouvrement de clé.

- étape 1 : retrouver la longueur  $\ell$  du mot clé
- étape 2 : effectuer  $\ell$  cryptanalyses de César sur les chiffrés décimés

Heuristique : les fréquences sur un **extrait** de français sont les mêmes que sur le français

# Chiffrements anciens

1 Substitutions alphabétiques

2 Transposition

## Exemple du lignes-colonnes

### Description

- On écrit le texte clair en lignes dans un tableau (nombre et ordre des colonnes secrets).

Clair : HERE IS A SECRET MESSAGE ENCIPHERED BY TRANSPOSITION

Clé : 1 10 7 11 3 8 6 4 9 2 5

Écriture en tableau :

Clé :	1	10	7	11	3	8	6	4	9	2	5
Clair :	H	E	R	E	I	S	A	S	E	C	R
	E	T	M	E	S	S	A	G	E	E	N
	C	I	P	H	E	R	E	D	B	Y	T
	R	A	N	S	P	O	S	I	T	I	O
	N										

- On lit le chiffré en colonnes, dans l'ordre secret.

Chiffré : HECRN CEYI ISEP SGDI RNTO AAES RMPN SSRO EEBT ETIA EEHS

Cryptanalyse : recouvrement de clé à clair connu

## Une propriété à retenir

Dans une substitution

- les symboles du texte clair sont **remplacés** ;
- les symboles du chiffré sont différents de ceux du clair (bien que même alphabet)

Dans une transposition

- les symboles du texte clair sont **déplacés** ;
- les symboles du chiffré sont les mêmes que ceux du clair

– Partie 2 –  
Attaquants. Modèles de sécurité

# Attaquants. Modèles de sécurité

1 Approche intuitive

2 Modèle d'attaquant

3 Modèles de sécurité

## Une première définition de la notion de sécurité

Un algorithme cryptographique est *sûr* lorsqu'un attaquant est incapable de deviner tout type d'information qu'il ne connaît pas : tout ou partie du clair, de la clé, etc. On parle de *sécurité sémantique*.

La sécurité sémantique est liée à la propriété des fonctions de chiffrement de "ressembler" à des fonctions aléatoires. Cela va plus loin que l'impossibilité de retrouver la clé.

## Les principes de Kerckhoffs

- ① Le système doit être matériellement, sinon mathématiquement indéchiffrable ;
- ② Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi ;
- ③ La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants ;
- ④ Il faut qu'il soit applicable à la correspondance télégraphique ;
- ⑤ Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes ;
- ⑥ Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.

# Attaquants. Modèles de sécurité

1 Approche intuitive

2 Modèle d'attaquant

3 Modèles de sécurité

## Attaque : un adversaire et un scénario

Un adversaire est caractérisé par :

- sa connaissance de la spécification de l'algorithme
- ses capacités : calcul, mémoire, action sur le canal
- un scénario, ou jeu, comprenant
  - un contexte d'apprentissage
  - un défi

## Complexité d'un algorithme

- Complexité temps : en nombre d'opérations élémentaires
- Complexité mémoire : en nombre d'octets

Avec l'état de l'art technologique 2020

facilement atteignable	$2^{40}$
limite atteignable	$2^{64}$
minimum inatteignable	$2^{80}$
raisonnablement inatteignable	$2^{128}$

## Les contextes de l'attaquant

- *chiffré connu* : observation seule des chiffrés sur la ligne ;
  - *clair connu* : acquisition de couples (clair, chiffré) par observation d'une machine de chiffrement ;
  - *clair choisi* : acquisition des couples (clair, chiffré) par accès à un *oracle* de chiffrement ;
  - *chiffré choisi* : acquisition de couples (clair, chiffré) par accès à un *oracle* de déchiffrement ;
- Souvent, “chiffré choisi” est un raccourci pour “clair et chiffré choisis”.

## Les défis

Par ordre décroissant de difficulté

- trouver la totalité de la clé : on parle alors de *cryptanalyse* ;
- trouver une partie de la clé ;
- déchiffrer un chiffré particulier ;
- trouver une information partielle sur un clair particulier ;

Un attaquant est dit *adaptatif* s'il a encore accès aux oracles une fois le défi posé par le challenger.

# Attaquants. Modèles de sécurité

- 1 Approche intuitive
- 2 Modèle d'attaquant
- 3 Modèles de sécurité

## Sécurité inconditionnelle

L'attaquant ne peut résoudre le défi quelles que soient ses capacités.

- p. ex., à chiffré connu, l'attaquant ne peut récupérer aucune information sur le clair ni sur la clé
- n'existe qu'en théorie, grâce au *One Time Pad* sous des hypothèses précises et un contexte très spécifique
- Peu pratique à formaliser et à manipuler, on lui préfère aujourd'hui la notion de *sécurité calculatoire*

## Sécurité calculatoire

Pour des raisons de facilité de mise en oeuvre (Kerckhoffs, principe 6), les algos modernes utilisent des objets (notamment clés) de taille finie.

- il y a toujours des attaques de complexité (temps, mémoire) finies, fonction des paramètres (taille clé, ...) de l'algorithme.  
Notamment la *brute force* et l'*attaque algébrique*
- Chaque attaque est imaginée et réalisée dans un contexte donné et pour un défi donné
- sa complexité temps/mémoire est calculée ou estimée sur le papier

En pratique :

- une attaque sera réalisable si sa complexité (calculée) est inférieure aux capacités technologiques de l'adversaire

**Attention.** Ne pas confondre :

- la complexité du calcul **légitime** du chiffrement/déchiffrement, qui doit être la plus insignifiante possible ;
- la complexité d'une **attaque contre l'algorithme** de chiffrement/déchiffrement, qui devra demeurer très grande/inaccessible en temps humain (l'attaquant la veut la plus faible possible).

## Le *niveau de sécurité*

- c'est la complexité de la meilleure attaque.
- il est à confronter au niveau de risque accepté et assumé par les utilisateurs légitimes :  
aujourd'hui, la valeur communément agréée pour le niveau de risque est de
  - $2^{128}$  si on parle de complexité
  - $2^{-128}$  si on parle de probabilité

La sécurité n'est pas une notion binaire (sûr/non sûr), mais quantitative (sûr jusqu'à un certain seuil)

Analogue de la résistance d'un objet à des contraintes mécaniques, électriques, thermiques, . . .

# Sécurité concrète et intrinsèque

Deux déclinaisons pour la notion de sécurité

- *Sécurité concrète* : le niveau de sécurité est supérieur à la limite technologique.  
Aucune attaque n'est faisable en pratique
- *Sécurité intrinsèque* : le niveau de sécurité est celui de la brute force  
Aucune des attaques intelligentes n'est meilleure que la meilleure attaque générique

## Méthodologie maximaliste

En général, on prend un modèle d'*attaquant fort*

- apprentissage le plus avantageux possible : typiquement clair et chiffré choisi ;
- défi le plus facile : typiquement distinguer la fonction d'une fonction aléatoire.

Ces attaquants ne sont pas forcément réalistes, mais sont plus puissants que les attaquants réalistes.

Philosophie du “qui peut le plus peut le moins” : un algorithme résistant à un attaquant fort résiste *a fortiori* à un attaquant réaliste.

ATTENTION : ne pas satisfaire les exigences les plus strictes ne signifie pas être faible !

## – Partie 3 – Chiffrement symétrique

## Définition et illustration

Un algorithme de chiffrement est dit *symétrique* ou *à clé secrète* lorsque la clé est une donnée secrète qu'Alice et Bob ont préalablement établie en commun.

- Alice veut envoyer un message à Bob, de manière sécurisée.
- Alice et Bob disposent d'une clé secrète commune .



## Mécanisme de chiffrement symétrique :

- Alice place le message dans un coffre, fermé par la clé commune,
- Bob ouvre le coffre avec la clé commune, et obtient le message.



Confidentialité : seuls Alice et Bob peuvent lire le message.

## Mécanisme de chiffrement symétrique :

- Alice place le message dans un coffre, fermé par la clé commune,
- Bob ouvre le coffre avec la clé commune, et obtient le message.



Confidentialité : seuls Alice et Bob peuvent lire le message.

## Mécanisme de chiffrement symétrique :

- Alice place le message dans un coffre, fermé par la clé commune,
- Bob ouvre le coffre avec la clé commune, et obtient le message.



Confidentialité : seuls Alice et Bob peuvent lire le message.

## Mécanisme de chiffrement symétrique :

- Alice place le message dans un coffre, fermé par la clé commune,
- Bob ouvre le coffre avec la clé commune, et obtient le message.



Confidentialité : seuls Alice et Bob peuvent lire le message.

## Mécanisme de chiffrement symétrique :

- Alice place le message dans un coffre, fermé par la clé commune,
- Bob ouvre le coffre avec la clé commune, et obtient le message.



**Confidentialité** : seuls Alice et Bob peuvent lire le message.

## Panorama des solutions

Techniques anciennes : opérations alphabétiques

- *substitutions*
- *transpositions*

Techniques modernes : opérations sur les bits

- le *One time Pad*, masque jetable
- le *chiffrement par flot* (*stream cipher*), encore appelé chiffrement par génération de pseudo aléa (GPA).
- le *chiffrement par bloc* (*block cipher*).

# Chiffrement symétrique

## 1 Chiffrement par flot

- L'approche Shannon
- Générateur de pseudo aléa
- Registres linéaires et fonctions booléennes
- Sécurité du chiffrement par flot
- Sécurité d'un GPA

## 2 Chiffrement par bloc

## 3 Sécurité d'un chiffrement symétrique

# Chiffrement symétrique

## 1 Chiffrement par flot

- L'approche Shannon
- Générateur de pseudo aléa
- Registres linéaires et fonctions booléennes
- Sécurité du chiffrement par flot
- Sécurité d'un GPA

# Chiffrement parfait

La définition qui suit est issue de la théorie de l'information.

On considérons un modèle où  $P$ ,  $C$ ,  $K$  sont des variables aléatoires.

Un chiffrement est dit *parfait* si, étant donné un texte chiffré  $C$ , et la clé  $K$  étant inconnue, l'incertitude sur le texte clair est la même que ce qu'elle était sans la connaissance du chiffré.

La distribution de probabilités du clair sachant le chiffré est égale à la distribution *a priori*

## Le chiffrement de Vernam, (*One Time Pad*, masque jetable)

- Alice et Bob possèdent une *suite chiffrante* secrète de la même longueur que le clair.
- Alice chiffre par XOR du clair et de la suite chiffrante.

Clair :	$p_0$	$p_1$	$p_2$	$p_3$	...	
$\oplus$	<b>Suite chiffrante :</b>	$z_0$	$z_1$	$z_2$	$z_3$	...
	Chiffré :	$c_0$	$c_1$	$c_2$	$c_3$	...

- Bob déchiffre identiquement. Opération *involutive*
- Chiffrement parfait si la suite chiffrante est une séquence aléatoire (théorème de Shannon)

**MAIS** la suite chiffrante

- doit être aussi longue que le message
- doit être utilisée une seule fois.

# Chiffrement symétrique

## 1 Chiffrement par flot

- L'approche Shannon
- Générateur de pseudo aléa
- Registres linéaires et fonctions booléennes
- Sécurité du chiffrement par flot
- Sécurité d'un GPA

## Génération de pseudo aléa

### Idée :

- imiter l'algorithme de Vernam
- avec une clé secrète courte et largement réutilisable

### Définition :

Un *générateur de pseudo aléa* (GPA) est un automate à états finis et à temps discret fournissant à partir d'un germe de taille faible, des suites déterministes indiscernables de suites aléatoires lorsqu'on ne connaît pas ce germe.

### En résumé :

Pseudo-aléa = déterministe *via* la connaissance du germe mais indiscernable d'aléa sinon.

## Chiffrement par flot avec un GPA

Le clair est toujours XORé bit à bit avec une “suite chiffrante”.

- Dans Vernam, la clé secrète **est** la suite chiffrante
- Avec un GPA, la clé secrète **est un germe qui contribue à générer** la suite chiffrante

Plus précisément : suite chiffrante = GPA(clé, marquant)

- Initialisation de l'état interne du GPA : copier la clé et le marquant
- Génération : à chaque coup d'horloge
  - produire un bit de pseudo aléa par une fonction  $\mathcal{F}$ ,
  - le xorer au bit courant de clair
  - mettre à jour l'état interne par une fonction bijective  $\mathcal{A}$

## Avec un dessin

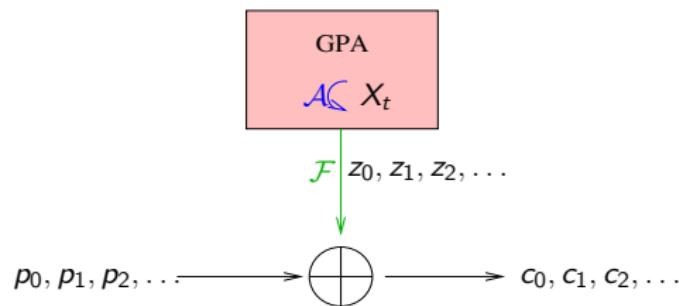
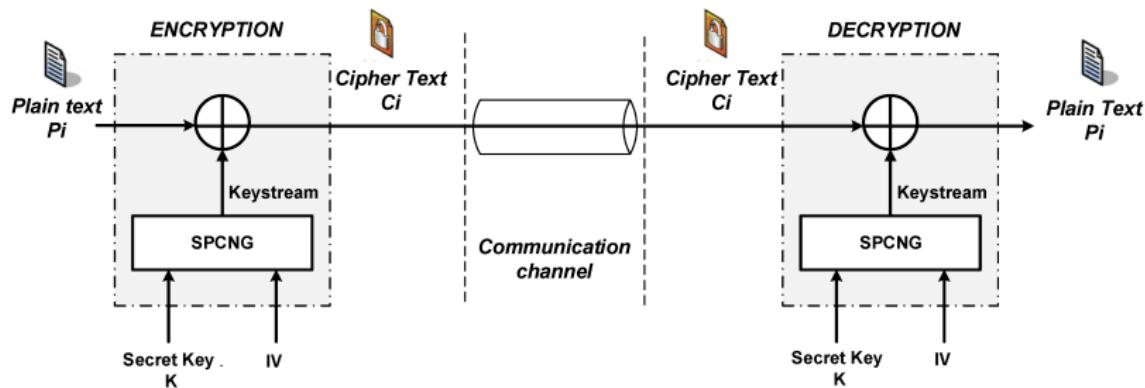


Figure – Chiffrement par flot par XOR avec une suite pseudo aléatoire

## Vue chiffrement et déchiffrement



## En équations

### Chiffrement du message binaire $(p_0, p_1, \dots)$ .

- 1 tirer au hasard  $\text{IV}$  /\* non prédictible et non reproductible \*/
- 2  $X_0 \leftarrow K, \text{IV}$  /\* initialise l'état interne du GPA \*/
- 3 Pour  $t = 0, 1, \dots$ 
  - 4  $z_t = \mathcal{F}(X_t)$  /\* Calcul d'un bit de pseudo aléa \*/
  - 5  $c_t = p_t \oplus z_t$  /\* xor avec le bit de clair \*/
  - 6  $X_{t+1} = \mathcal{A}(X_t)$  /\* Mise à jour de l'état interne \*/
- 7 fin pour
- 8 Alice transmet  $(\text{IV}, c_0, c_1, \dots)$  à Bob.

## Le rôle du marquant

**Chiffrement** d'un autre message binaire  $(p'_0, p'_1, \dots)$ .

- 1 tire  $\textcolor{blue}{IV'}$  /\*  $\neq IV$  \*/
- 2  $X'_0 \leftarrow K, \textcolor{blue}{IV'}$  /\* initialise l'état interne du GPA \*/
- 3 Pour  $t = 0, 1, \dots$ 
  - 4  $z'_t = \mathcal{F}(\textcolor{blue}{X}'_t)$  /\* Calcul d'un bit de pseudo aléa \*/
  - 5  $c'_t = p'_t \oplus z'_t$  /\* xor avec le bit de clair \*/
  - 6  $\textcolor{blue}{X}'_{t+1} = \mathcal{A}(\textcolor{blue}{X}'_t)$  /\* Mise à jour de l'état interne \*/
- 7 fin pour
- 8 Alice transmet  $(\textcolor{blue}{IV'}, c'_0, c'_1, \dots)$  à Bob.

## Propriétés liées au marquant

- le marquant est un complément d'initialisation devant changer à chaque nouveau chiffrement
- $(z'_t)$  est différente (et même statistiquement indépendante) de  $(z_t)$  : critère de Vernam satisfait !

L'algorithme de chiffrement est *randomisé* : tirage et utilisation d'une variable auxiliaire aléatoire

- conséquence “originale” :

plusieurs chiffrements du même texte clair  $\Rightarrow$  plusieurs marquants  $\Rightarrow$  plusieurs suites chiffrantes  $\Rightarrow$  plusieurs chiffrés différents !

- cette originalité s'avèrera en fait cruciale pour les critères modernes de sécurité.

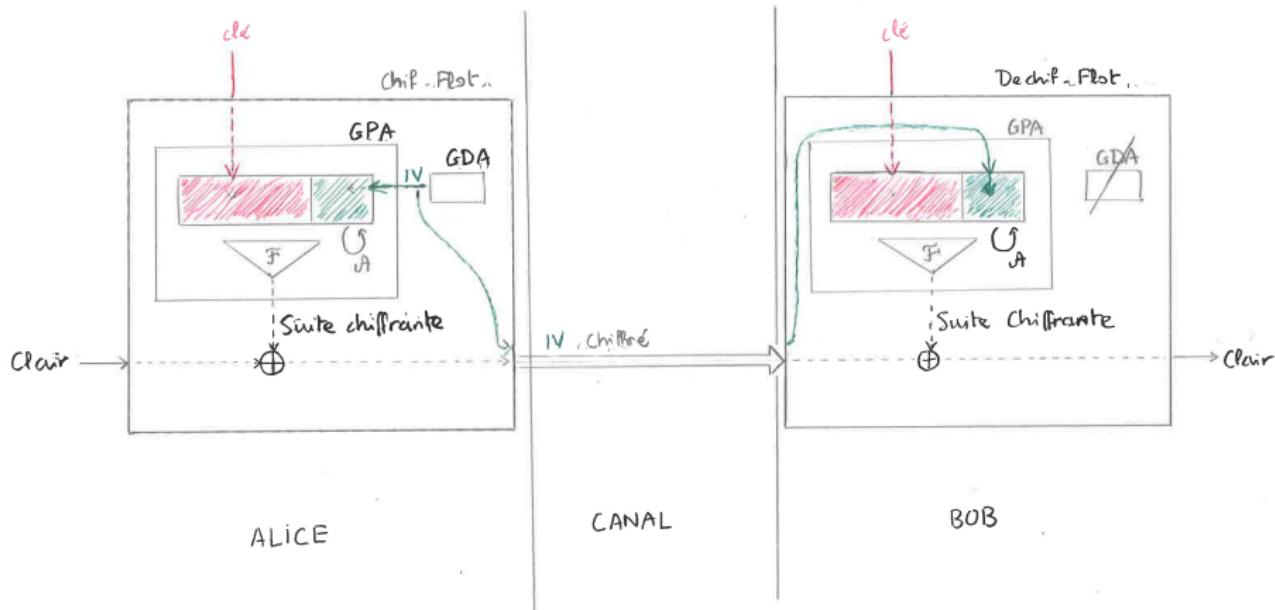
**Remarque.** Le déchiffrement n'est pas randomisé :

- Bob ne doit pas tirer un IV aléatoire, mais utiliser celui envoyé par Alice
- le déchiffrement annule/compense la randomisation du chiffrement
- Bob retrouve le même texte clair quel que soit le marquant utilisé par Alice

## Déchiffrement du message binaire ( $\text{IV}, c_0, c_1, \dots$ ).

- ①  $X_0 \leftarrow K, \text{IV}$  /\* initialise l'état interne du GPA \*/
- ② Pour  $t = 0, 1, \dots$ 
  - ③  $z_t = \mathcal{F}(X_t)$  /\* Calcul d'un bit de pseudo aléa \*/
  - ④  $p_t = c_t \oplus z_t$  /\* xor avec le bit de chiffré \*/
  - ⑤  $X_{t+1} = \mathcal{A}(X_t)$  /\* Mise à jour de l'état interne \*/
- ⑥ fin pour
- ⑦ Bob retourne  $(p_0, p_1, \dots)$ .

## Schéma de synthèse



## Tailles typiques des paramètres

Clé : de 128 à 256 bits

Marquant (*IV*) : de 64 à 128 bits

État interne :  $> \max(|\text{clé}| + |\text{marquant}|, 2 \times |\text{clé}|)$

Suites chiffrantes : longueur quelconque (c'est fait pour)

# Chiffrement symétrique

## 1 Chiffrement par flot

- L'approche Shannon
- Générateur de pseudo aléa
- Registres linéaires et fonctions booléennes
- Sécurité du chiffrement par flot
- Sécurité d'un GPA

## Quels ingrédients pour construire un GPA ?

On combine

- automates de grande période
- fonctions booléennes non linéaires (polynômes de degré > 1)

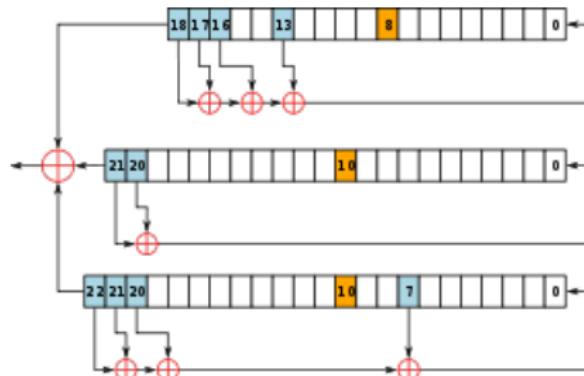
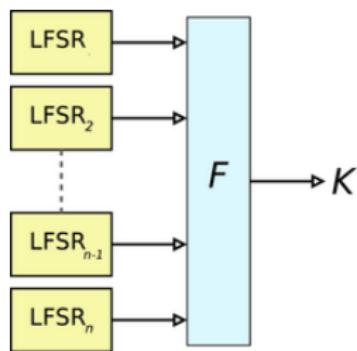


Figure – Automates linéaires filtrés par une fonction non linéaire

Figure – Algorithme A5-1 pour les mobiles

# LFSR, *Linear Feedback Shift Register*

*Registre à décalage à rétroaction linéaire*

C'est un automate à temps discret défini par :

- un état interne  $X_t = (x_{0,t}, x_{1,t}, \dots, x_{L-1,t})$  de  $L$  bits ;
- une fonction d'avance de la forme :

pour  $t \geq 0$ ,

$$\begin{cases} x_{0,t+1} &= x_{1,t} \\ x_{1,t+1} &= x_{2,t} \\ \dots \\ x_{L-2,t+1} &= x_{L-1,t} \\ x_{L-1,t+1} &= a_0 x_{0,t} \oplus a_1 x_{1,t} \oplus \dots \oplus a_{L-1} x_{L-1,t} \end{cases}$$



FIGURE. – Schéma synoptique d'un LFSR

Les coefficients  $a_0, \dots, a_{L-1} \in \mathbb{F}_2$  caractérisent la rétroaction. On les regroupe formellement dans le *polynôme de rétroaction*

$$P(T) = a_0 \oplus a_1 T \oplus \dots \oplus a_{L-1} T^{L-1} \oplus T^L \in \mathbb{F}_2[T]$$

## Fonctions booléennes

- On appelle fonction booléennes de  $n$  variables toute application de  $\mathbb{F}_2^n \rightarrow \mathbb{F}_2$ .
- Représentation combinatoire par la table de valeurs ;
- représentation algébrique par un polynôme binaire à  $n$  variables, de degré  $0 \leq d \leq n$  (NB : en contexte crypto  $2 \leq d \leq n - 1$ )

# Chiffrement symétrique

## 1 Chiffrement par flot

- L'approche Shannon
- Générateur de pseudo aléa
- Registres linéaires et fonctions booléennes
- **Sécurité du chiffrement par flot**
- Sécurité d'un GPA

## Validation modulaire de la sécurité

Le principe du chiffrement par XOR :

$$\text{chiffré} = \text{clair} \oplus \text{suite chiffrante}$$

a été validé une fois pour toutes par Vernam.

Il reste évaluer intrinsèquement la qualité d'un GPA. Pour ce faire, on suppose connue la suite chiffrante (incluant le marquant)

# Chiffrement symétrique

## 1 Chiffrement par flot

- L'approche Shannon
- Générateur de pseudo aléa
- Registres linéaires et fonctions booléennes
- Sécurité du chiffrement par flot
- Sécurité d'un GPA

## Qu'apporte chaque ingrédient ?

Les éléments linéaires (e.g. LFSR) produisent des *suites récurrentes linéaires* :

$$\forall t \geq 0, \quad a_0 s_t + a_1 s_{t+1} + \cdots + a_{L-1} s_{t+L-1} + s_{t+L} = 0$$

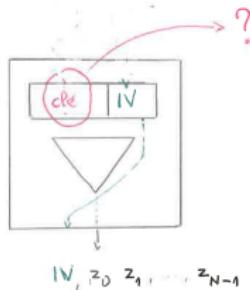
aux propriétés

- elles sont périodiques, de période divisant  $2^L - 1$
- on sait prescrire la période maximale =  $2^L - 1$  par un bon choix de conception prendre  $P(T)$  un polynôme *primitif*.
- si elles sont observées sur une taille  $\ll$  leur période, elles **ressemblent à des suites aléatoires**
- leurs propriétés statistiques sont très bonnes
- (**Problème**) elles dépendent linéairement des bits de clé  $\Rightarrow$  attaque algébrique faisable (cf. *infra*)

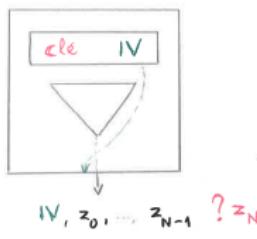
La composition d'un élément linéaire par une fonction booléenne de degré  $d$  assez grand ( $a_{minima} > 1$ ) permet de :

- conserver les propriétés souhaitables des suites produites : grandes périodes, statistiques pratiquement idéales ;
- supprimer l'inconvénient de la linéarité : les suites produites sont de degré  $d$  en les bits de clé  $\Rightarrow$  attaque algébrique pratiquement infaisable.

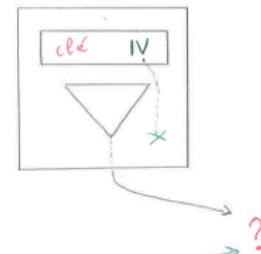
# Typologie classique des attaques (en dessin)



Recouvrement de clé



Prédiction du bit suivant



Indistinguisabilité  
Real - Random



## Typologie classique des attaques (détails)

### Attaques en recouvrement de clé

Connaissant une ou plusieurs séquences pseudo aléatoires (incluant l'IV)  $(IV, z_0, z_1, \dots)$ , retrouver  $K$ .

- Brute force :  $2^{|K|}$ .
- Attaque algébrique :

$$\approx \left( \sum_{i=0}^d \binom{|K|}{i} \right)^3, \quad \approx O(|K|^{3d}) \text{ quand } d \ll |K|,$$

- compromis temps-mémoire sur l'état interne (cf. exercice) :  $2^{|X|/2}$
- Attaque par corrélation entre suite chiffrante et état interne (cf. exercice) :  $O(1/\varepsilon^2)$  où  $\varepsilon$  est un biais de corrélation.

## Évaluation du caractère (soi-disant) aléatoire

- ① Attaque par prolongement (prédiction du bit suivant)  
Connaissant une séquence particulière  $(IV, z_0, z_1, \dots, z_{N-1})$ , deviner le prochain bit  $z_N$
- ② Attaque en distinguateur *real-random*  
Étant donné une suite  $(\tilde{z}_0, \tilde{z}_1, \dots, \tilde{z}_{N-1})$ , dire si cette suite est parfaitement aléatoire ou si elle issue du GPA.

Dans les deux cas

- ① Attaques
  - Brute force :  $2^{|K|}$ .
  - statistiques, par exemple exploitation d'un biais  $\varepsilon$  d'équidistribution :  $O(1/\varepsilon^2)$
- ② La probabilité de succès ne doit pas être (significativement) éloignée de  $1/2$  pour toute attaque de complexité humaine

## Indistinguabilité “real-random”

Un attaquant  $A$  voit une suite parfaitement aléatoire ou issue du GPA. Il doit deviner la situation. On parle de *distinguer real-or-random*

Efficacité de l'attaquant  $A$  mesurée par l'*avantage* :

$$\begin{aligned}\mathbf{Adv}^{rr}(A) &= \Pr(A \text{ prétend “réel”} \mid \text{réel}) \\ &\quad - \Pr(A \text{ prétend “réel”} \mid \text{aléatoire}) \\ &= 2 \left( \Pr(A \text{ devine bien}) - \frac{1}{2} \right) \\ &\in [-1, 1]\end{aligned}$$

## L'avantage

- est calculé comme une fonction des paramètres du GPA et de la taille de la suite chiffrante connue ;
- s'il est  $\leq 2^{-20}$ , le GPA est dit indistinguable *real-random*

## Synthèse

Sécurité	Attaquant	Probabilité $p$	Avantage
Parfaite	Au hasard	$1/2$	0
Oui	Inefficace	$ p - 1/2  \leq 2^{-21}$	valeur absolue $\leq 2^{-20}$
Non	Efficace	$ p - 1/2  > 2^{-21}$	valeur absolue $> 2^{-20}$

# Chiffrement symétrique

## 1 Chiffrement par flot

## 2 Chiffrement par bloc

- Primitive bloc
- Les ingrédients d'une primitive bloc
- Sécurité d'une primitive bloc
- Mode d'opération

## 3 Sécurité d'un chiffrement symétrique

# Chiffrement par bloc

Un algorithme bloc s'appuie sur

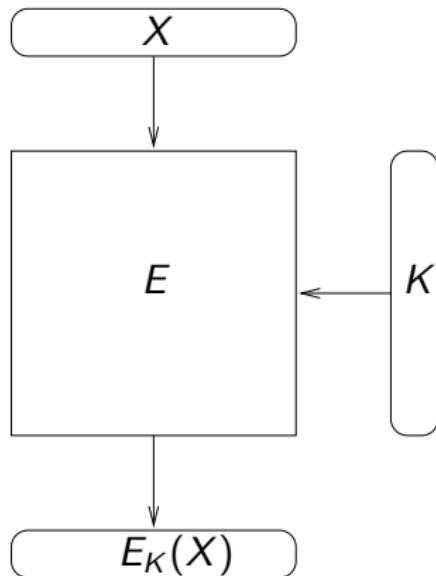
- l'utilisation d'une *primitive bloc*  
famille  $E_K$  de permutations de  $\{0, 1\}^b$  paramétrée par la clé  
 $K \in \{0, 1\}^k$ ,  
 $b$  taille du bloc,  $k$  taille de la clé sont fixées
- la spécification de *modes d'opération*  
sur couche algorithmique faisant appel à la primitive comme  
une sous routine

# Chiffrement symétrique

## 2 Chiffrement par bloc

- **Primitive bloc**
- Les ingrédients d'une primitive bloc
- Sécurité d'une primitive bloc
- Mode d'opération

## Définition d'une primitive bloc

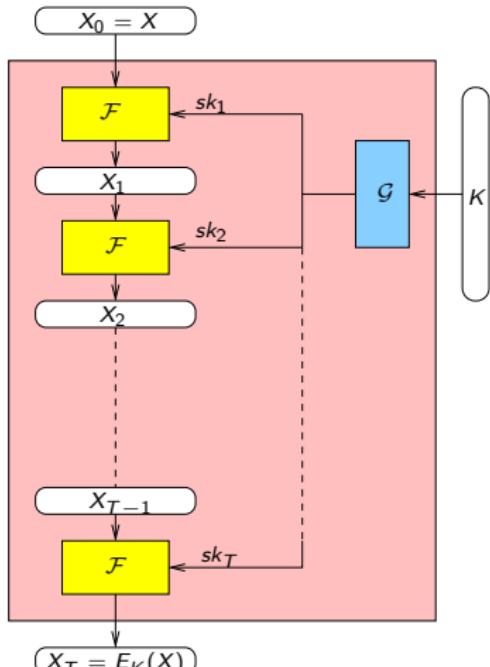


$E$  *primitive bloc* : famille de permutations sur  $\{0, 1\}^b$ ,  $X \mapsto Y$  paramétrée par la clé  $K \in \{0, 1\}^k$   
 $(K, X) \mapsto E_K(X)$

- $b$  est la taille du bloc
- $k$  est la taille de la clé
- $X$  est le bloc d'entrée (le “clair”)
- $Y = E_K(X)$  est le bloc de sortie (le “chiffré”)

Figure – Schéma général d'un algorithme de chiffrement par bloc

## Schéma itératif



- $T$  nombre de tours
- $\mathcal{F}$  fonction de tour :  
 $(sk_i, X_{i-1}) \mapsto X_i$
- $\mathcal{G}$  fonction génération de sous-clés :  
 $K \mapsto (sk_1, \dots, sk_T)$

## Les caractéristiques d'une primitive bloc

La spécification d'une primitive contient explicitement :

- la taille  $b$  du bloc  $X$  (et du bloc  $Y$ ),
- la taille  $k$  de la clé  $K$ ,
- le nombre de tours  $T$ ,
- la description des composantes élémentaires,
- la description de la fonction  $\mathcal{F}$  de tour ,
- la description de la fonction  $\mathcal{G}$  de génération de sous-clés.  
on dit aussi “diversification de clé” ou “cadencement de clé”.

## Taille et choix des paramètres

Clé : de 128 à 256 bits

Bloc : de 128 à 256 bits

Sous-clé : égale à ou de l'ordre de la taille du bloc

# Chiffrement symétrique

## 2 Chiffrement par bloc

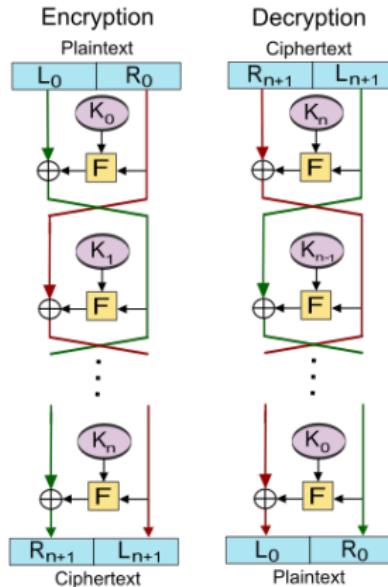
- Primitive bloc
- **Les ingrédients d'une primitive bloc**
- Sécurité d'une primitive bloc
- Mode d'opération

## Les ingrédients mathématiques

Comme composantes élémentaires pour fabriquer les primitives blocs, se trouvent :

- fonctions linéaires : transpositions, rotations, applications matricielles, ...
- fonctions de substitutions non linéaires données par leur table de valeurs ou leur expression algébrique (polynomiale multivariée)

## Exemples de constructions (1/3). Schéma de Feistel

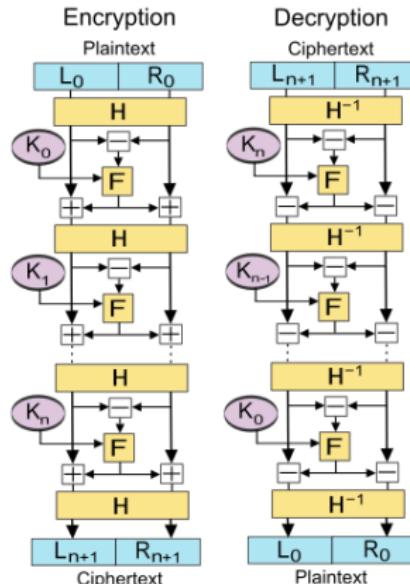


## Schéma de Feistel : exemple du DES

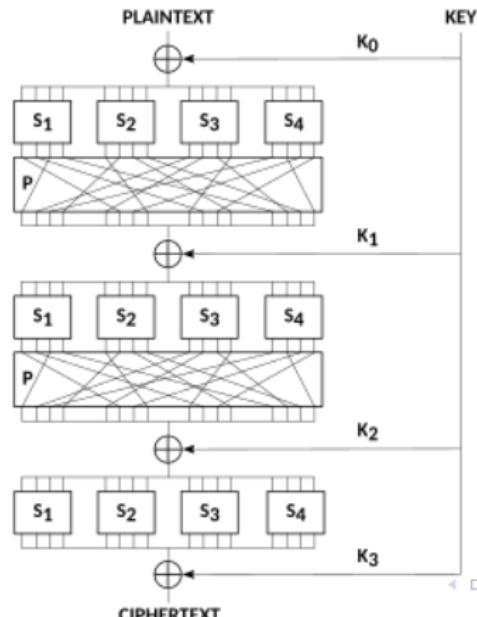
DES (*Data Encryption Standard*), standard jusqu'en 2000.

- conception fin des années 60
- bloc 64 bits, clé 56 bits
- 16 tours de *Feistel* :
  - la variable de tour  $X_i$  est coupée en une partie gauche  $L_i$  et une partie droite  $R_i$
  - $K_i$  est la clé de tour
  - $L_{i+1} = R_i$ ,  $R_{i+1} = L_i \oplus F_{K_i}(R_i)$ ,
  - $F_{K_i}$  pas forcément bijective.

## Exemples de constructions (2/3). Diagramme de Lai-Massey



## Exemples de constructions (3/3). Substitutions-Permutations



# L'Advanced Encryption Standard (AES)

Issu d'une compétition ouverte (1998-2000). Cahier des charges :

- taille des blocs : 128 bits ;
- taille des clés : 128, 192 ou 256 bits
- compromis d'efficacité en logiciel et en matériel
- résistant à toutes les attaques connues alors

Le vainqueur : Rijndael (Rijmen, Daemen)

- Schéma itératif de type SPN. Nombre de tours : 10, 12, 14 pour les 3 tailles de clé
- ingrédients basés sur une structure algébrique forte définie sur les octets.
- propose en "bonus" (non retenus dans le standard) des versions pour tailles de clés et de blocs : 128, 160, 192, 224, 256 bits

## Un petit coup de loupe sur l'AES-128

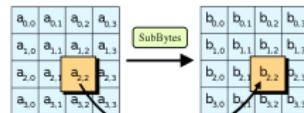
Taille de bloc = taille de clé = 128 bits

Structure de la variable de bloc (données et sous-clés)

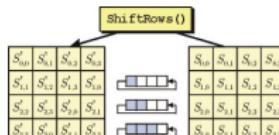
$a_0$	$a_4$	$a_8$	$a_{12}$	=	$s_{0,0}$	$s_{0,1}$	$s_{0,2}$	$s_{0,3}$
$a_1$	$a_5$	$a_9$	$a_{13}$		$s_{1,0}$	$s_{1,1}$	$s_{1,2}$	$s_{1,3}$
$a_2$	$a_6$	$a_{10}$	$a_{14}$		$s_{2,0}$	$s_{2,1}$	$s_{2,2}$	$s_{2,3}$
$a_3$	$a_7$	$a_{11}$	$a_{15}$		$s_{3,0}$	$s_{3,1}$	$s_{3,2}$	$s_{3,3}$

- Les  $a_i$  (et les  $s_{i,j}$ ) sont des octets, i.e.  $\in \{0,1\}^8$
- $\{0,1\}^8$  est muni d'une structure de corps, noté  $\mathbb{F}_{2^8}$  permettant les notions et opérations bien connues dans  $\mathbb{R}$  : vecteurs, matrices, addition, multiplication, algèbre linéaire, ...

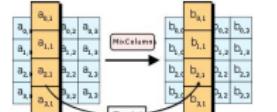
# Les fonctions élémentaires



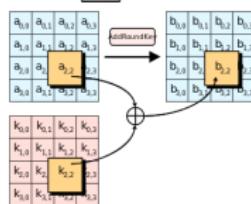
- SubBytes()



- ShiftRows()



- MixColumns()



- AddRoundKey()

## AES-128 : le déroulé complet

- Tour initial (tour 0)
  - AddRoundKey()

Clé du tour 0 := clé maîtresse
- Tours intermédiaires complets (tours 1 à 9) :
  - SubBytes()
  - ShiftRows()
  - MixColumns()
  - AddRoundKey()
- Tour final (tour 10)
  - SubBytes()
  - ShiftRows()
  - AddRoundKey()

# Chiffrement symétrique

## 2 Chiffrement par bloc

- Primitive bloc
- Les ingrédients d'une primitive bloc
- Sécurité d'une primitive bloc
- Mode d'opération

## Sécurité d'une primitive bloc : les fondamentaux

La famille  $(E_K, E_K^{-1})$  doit :

- être robuste au recouvrement de clé à clairs et chiffrés choisis : connaissant des  $(X, E_K(X))$  et  $(Y, E_K^{-1}(Y))$ , retrouver  $K$  est pratiquement impossible et coûte  $\geq 2^{|K|}$ .
- assurer la *diffusion*  
chaque bit d'entrée influence tous les bits de sortie. Assuré par les éléments linéaires.
- assurer la *confusion*  
chaque bit de sortie est une expression algébrique de haut degré des bits d'entrées. Assuré par les éléments non linéaires.

## Sécurité d'une primitive bloc : IND-PRF

La famille  $(E_K, E_K^{-1})$  doit être indistinguable d'une famille de permutations aléatoires

- Pour  $K$  fixée et inconnue,  $E_K$  et  $E_K^{-1}$  doivent ressembler à des permutations aléatoires :
  - connaissant  $(X_1, E_K(X_1)), \dots, (X_s = E_K(X_s))$ , et étant donné un nouveau  $X_{s+1}$ , impossible de prédire  $E_K(X_{s+1})$  (mieux que par hasard)
  - connaissant  $(Y_1, E_K^{-1}(Y_1)), \dots, (Y_s = E_K^{-1}(Y_s))$ , et étant donné un nouveau  $Y_{s+1}$ , impossible de prédire  $E_K^{-1}(Y_{s+1})$  (mieux que par hasard)
- Les permutations sont indépendantes les unes des autres :
  - les connaissances accumulées concernant les clés passées  $K_1, \dots, K_s$  ne donne aucune information sur la clé du présent  $K_{s+1}$

## Dans l'écriture algébrique

**Exemple jouet fictif**  $E_{(k_0, \dots, k_7)} : (p_0, \dots, p_7) \mapsto (c_0, \dots, c_7)$

$$\begin{aligned}c_0 &= \cdots \oplus p_0 k_0 p_3 k_3 k_6 \oplus p_0 k_0 p_3 p_5 k_5 p_6 k_7 \oplus p_0 k_0 p_3 p_5 k_5 p_6 \oplus p_0 k_0 p_3 p_5 k_5 k_7 \oplus \cdots \\c_1 &= \cdots \oplus p_0 p_1 p_2 p_3 k_6 \oplus p_0 p_1 p_2 k_3 k_4 k_5 p_7 \oplus p_0 p_1 p_2 k_3 k_4 k_5 \oplus \cdots \\c_2 &= \cdots \oplus p_3 k_3 k_4 p_6 k_6 \oplus p_0 k_1 p_3 k_3 k_4 p_6 \oplus p_0 k_1 p_3 k_3 k_4 k_6 \oplus p_0 k_1 p_3 k_3 p_5 p_6 k_6 \oplus \cdots \\c_3 &= \cdots \oplus p_6 \oplus p_0 k_3 p_5 k_6 p_7 \oplus p_0 k_3 p_5 p_7 \oplus p_0 k_3 p_5 \oplus p_0 k_3 k_5 p_6 k_6 \oplus p_0 k_3 k_5 p_6 p_7 \oplus \cdots \\c_4 &= \cdots \oplus k_0 k_2 p_3 p_4 p_5 p_6 \oplus k_0 k_2 p_3 p_4 k_5 p_6 \oplus k_0 k_2 p_3 p_4 p_6 \oplus k_0 k_2 p_3 p_4 k_7 \oplus k_0 k_2 p_3 p_4 \oplus \cdots \\c_5 &= \cdots \oplus p_1 p_3 p_4 k_4 k_5 k_6 \oplus p_1 p_3 p_4 k_4 k_5 \oplus p_1 p_3 p_4 k_4 p_6 k_6 k_7 \oplus p_1 p_3 p_4 k_4 p_6 k_6 \oplus \cdots \\c_6 &= \cdots \oplus k_2 p_4 k_4 p_5 p_7 k_7 \oplus p_2 k_2 p_4 k_4 p_5 p_7 \oplus p_2 k_2 p_4 k_4 p_5 k_7 \oplus p_2 k_2 p_4 k_4 k_5 p_7 k_7 \oplus \cdots \\c_7 &= \cdots \oplus p_3 p_4 k_4 k_6 k_7 \oplus p_3 p_4 k_4 \oplus p_3 p_4 p_5 k_5 p_6 k_6 k_7 \oplus p_3 p_4 p_5 k_5 p_6 k_6 \oplus p_3 p_4 p_5 \oplus \cdots\end{aligned}$$

Chaque expression  $c_i = \text{polynome}(k_0, \dots, k_7, p_0, \dots, p_7)$  :

- diffusion : contient toutes les variables  $k_i$  et  $p_i$  ;
- confusion : est de haut degré par rapport aux variables  $k_i$  et  $p_i$ .

## Attaques classiques (en recouvrement de clé)

- Recherche exhaustive : complexité en temps  $2^{|clé|}$ .
- Attaques algébriques : généralement de complexité impraticable et non pertinente, car  $\gg 2^{|clé|}$
- (NB : pseudo-recouvrement de clé) Attaque par *dictionnaire* : compilation exhaustive des couples (entrée,sortie). Complexité en temps et en espace  $2^{|bloc|}$ .

# Deux incontournables : attaque différentielle et attaque linéaire

- Attaque différentielle
  - clair choisi
  - exploite un biais différentiel imperfection statistique / comportement non aléatoire uniforme dans la propagation des différences
  - complexité croissante avec le nombre de tours, jusqu'à un seuil
- Attaque linéaire
  - clair connu
  - exploite un biais linéaire corrélation entre bits de clé, clair, chiffré, c'est-à-dire égalité entre deux combinaisons linéaires de bits soit disant aléatoires n'ayant pas lieu avec probabilité 1/2
  - complexité croissante avec le nombre de tours, jusqu'à un seuil

# Attaque différentielle (Biham, Shamir, 1990)

## Propagation des différences.

Clairs	$P$	$\xrightarrow{E_K}$	$C$	Chiffrés
	$P'$		$C'$	
Différence	$\Delta P$ $= P \oplus P'$	$\xrightarrow{\mathbb{P}(\delta_{in} \rightarrow \delta_{out})}$	$\Delta C$ $= C \oplus C'$	

## Probabilité différentielle :

$$\mathbb{P}(\delta_{in} \rightarrow \delta_{out}) \stackrel{\text{def}}{=} \mathbb{P}(\Delta C = \delta_{out} | \Delta P = \delta_{in})$$

- si  $E_K$  est une fonction (pseudo) aléatoire,  
 $\forall \delta_{in}, \delta_{out}, \quad \mathbb{P}(\delta_{in} \rightarrow \delta_{out}) \approx 2^{-|\text{bloc}|}$
- s'il existe  $\delta_{in}, \delta_{out}$  tels que  $\mathbb{P}(\delta_{in} \rightarrow \delta_{out}) \gg 2^{-|\text{bloc}|}$ , on a trouvé un *biais différentiel* pour  $E_K$

## Phase théorique : préparation de l'attaque

- Détermination d'un biais différentiel sur  $T - 1$  tours, indépendant de la clé

$$\begin{array}{ccccc} P & \xrightarrow{sk_1, \dots, sk_{T-1}} & X_{T-1} & \xrightarrow{sk_T} & C \\ \delta_{in} & \xrightarrow{\quad} & \delta_{out} & & \\ p := \mathbb{P}(\delta_{in} \rightarrow \delta_{out}) & & & & \end{array}$$

- On fixe désormais  $\delta_{in}$  et  $\delta_{out}$

## Phase opérationnelle : réalisation de l'attaque

- Clair choisi
- Cible : la sous-clé  $sk_T$  du dernier tour
- Méthode : pari et test d'hypothèse

① Choix d'une collection  $\{P_i\}$  de  $O(1/p)$  textes clairs

② Obtention des trois collections

$$\{P'_i\} = \{P_i \oplus \delta_{in}\}, \{C_i = E_K(P_i)\}, \{C'_i = E_K(P'_i)\}$$

③ Pari  $sk_T$  sur la valeur de  $sk_T$

④ calcul de deux collections

$$\{X_{T-1,i} = \text{Tour}^{-1}(sk_T, C_i)\}, \{X'_{T-1,i} = \text{Tour}^{-1}(sk_T, C'_i)\}$$

⑤ Estimation statistique de la probabilité différentielle sur  $T - 1$  tours

$$\mathbb{P}(\Delta X_{T-1} = \delta_{out} | \Delta P = \delta_{in})$$

⑥ si  $\approx p$  alors pari correct, i.e.  $sk_T = sk_T$

⑦ sinon pari incorrect, i.e.  $sk_T \neq sk_T$

Situation réelle	$P, P'$ $\delta_{in}$	$\xrightarrow{sk_1, \dots, sk_{T-1}}$ $\xrightarrow{p}$	$X_{T-1}, X'_{T-1}$ $\Delta X_{T-1}$	$\xrightarrow{sk_T}$	$C, C'$
Situation pariée (pari bon/mauvais)	$P, P'$ $\delta_{in}$	$\xrightarrow{\mathbb{P}(\Delta X_{T-1} = \delta_{out}   \Delta P = \delta_{in})}$ $(\approx p / \approx 2^{- bloc })$	$X_{T-1}, X'_{T-1}$ $\Delta X_{T-1}$	$\xleftarrow{sk_T}$	$C, C'$

# Attaque linéaire (Matsui, 1994)

## Approximation linéaire

Égalité probabiliste entre combinaisons linéaires de clair chiffré clé

$$\alpha P \oplus \beta C \stackrel{p_{\alpha,\beta,\gamma}}{=} \gamma K$$

$$[P_{i_1} \oplus \cdots \oplus P_{i_a}] \oplus [C_{j_1} \oplus \cdots \oplus C_{j_b}] \stackrel{p_{\alpha,\beta,\gamma}}{=} [K_{\ell_1} \oplus \cdots \oplus K_{\ell_c}]$$

Pour  $\alpha, \beta, \gamma$  fixés :

- si  $E_K$  est une fonction (pseudo aléatoire),  $p_{\alpha,\beta,\gamma} = 1/2$
- s'il existe  $\alpha, \beta, \gamma$  tel que  $p_{\alpha,\beta,\gamma} \neq 1/2$ , on a trouvé un *approximation linéaire biaisée* (ou *biais linéaire*) pour  $E_K$

# 1re attaque

## Phase théorique. Préparation de l'attaque

- Détermination d'une approximation linéaire de  $E_K$  :  $\alpha, \beta, \gamma$ , de probabilité  $p \neq 1/2$ .

## Phase opérationnelle Réalisation de l'attaque

- Contexte clair connu
- Cible : la valeur de la combinaison linéaire  $\gamma K$ , i.e. une équation linéaire en les bits de  $K$
- Méthode : estimation et vote majoritaire

- Observation d'une collection  $\{P_i, C_i\}$  de  $O(1/|p_{\alpha,\beta,\gamma} - 1/2|^2)$  couples clairs/chiffrés
- Calcul de la collection des valeurs  $\{\alpha P_i \oplus \beta C_i\}$ . Soit *maj* la valeur majoritaire. On récupère l'équation linéaire en la clé :

	$p < 1/2$	$p > 1/2$
$maj = 0$	$\gamma K = 1$	$\gamma K = 0$
$maj = 1$	$\gamma K = 0$	$\gamma K = 1$

Recommencer avec d'autres approximations linéaires biaisées pour obtenir un système d'équations linéaires en les bits de clé

## 2me attaque

### Phase théorique. Préparation de l'attaque

- Détermination d'une approximation linéaire sur  $T - 1$  tours :  $\alpha, \beta, \gamma$ , de probabilité  $p^{[T-1]} \neq 1/2$ .

$$\alpha P \oplus \beta X_{T-1} = \gamma K$$

### Phase opérationnelle Réalisation de l'attaque

- Contexte clair connu
- Cible : la sous clé  $sk_T$  du dernier tour et la valeur de la combinaison linéaire  $\gamma K$ , i.e. une équation linéaire en les bits de  $K$
- Méthode : pari et test d'hypothèse ; estimation et vote majoritaire

- Observation d'une collection  $\{P_i, C_i\}$  de  $O(1/|p^{[T-1]} - 1/2|^2)$  couples clairs/chiffrés
- pari  $sk_T$  sur la dernière sous-clé  $sk_T$
- Calcul de la collection  $\{X_{T-1,i} = \text{Tour}^{-1}(sk_T, C_i)\}$
- Calcul de la collection des valeurs  $\{\alpha P_i \oplus \beta X_{T-1,i}\}$ ,
- si la distribution 0/1 est biaisée de probabilité  $p^{[T-1]} / 1 - p^{[T-1]}$  et de valeur majoritaire  $maj$ , le pari est bon et l'équation linéaire cherchée est

	$p^{[T-1]} < 1/2$	$p^{[T-1]} > 1/2$
$maj = 0$	$\gamma K = 1$	$\gamma K = 0$
$maj = 1$	$\gamma K = 0$	$\gamma K = 1$

- si la distribution 0/1 est équilibrée, le pari est mauvais et on ne récupère pas d'équation linéaire
- en pratique, la distribution expérimentale 0/1 est déséquilibrée pour plusieurs (voire tous les) paris. Le bon pari est identifié comme celui provoquant le déséquilibre maximal.

On peut préparer plusieurs approximations linéaires biaisées sur  $T - 1$  tours : lorsque le pari est bon, on récupère plusieurs équations linéaires en la clé.

# Chiffrement symétrique

## 2 Chiffrement par bloc

- Primitive bloc
- Les ingrédients d'une primitive bloc
- Sécurité d'une primitive bloc
- Mode d'opération

# Mode d'opération d'un chiffrement par bloc

## Position du problème

- On veut traiter des clairs de taille quelconque
- on dispose d'une primitive bloc (taille fixe)

$$E_K : \{0, 1\}^b \rightarrow \{0, 1\}^b, K \in \{0, 1\}^k$$

## Principe

- Découper le clair en  $\ell$  blocs  $P_1, \dots, P_{\ell-1}, P_\ell$
- utiliser répétitivement la primitive  $E_K$  pour traiter chacun des blocs

## Faut-il $|P|$ multiple de $b$ ?

- Certains modes traitent nativement le cas  $|P|$  non multiple de  $b$ .

On notera ici  $P_1, \dots, P_{\ell-1}, P_\ell^*$ , avec  $P_\ell^*$  possiblement incomplet, c'est-à-dire  $|P_1| = \dots = |P_{\ell-1}| = b$ ,  $1 \leq |P_\ell^*| \leq b$

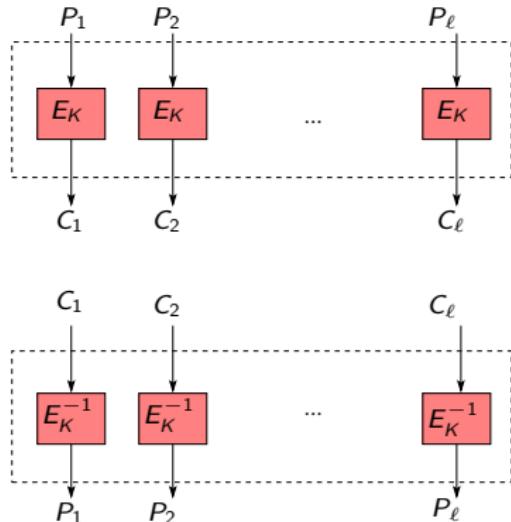
- D'autres modes nécessitent que  $|P|$  soit multiple de  $b$ .

L'algorithme de chiffrement applique un prétraitement (à prévoir par le concepteur), le *padding* : allongement du clair  $P$  en  $P_{\text{pad}}$  par des bits finaux non significants de sorte que  $|P_{\text{pad}}|$  soit multiple de  $b$ .

## Les modes de fonctionnement classiques

- Les modes blocs “purs et durs”
  - ECB, *Electronic Code Book* : appelé *mode dictionnaire*
  - CBC, *Cipher Block Chaining*, mode dit “chaîné”, i.e. le calcul du bloc chiffré courant dépend des valeurs de chiffrés calculées précédemment.
- Les chiffrement par flot déguisés en bloc (ou le contraire !)
  - CTR, *Counter*. Parallélisable.
  - OFB, *Output Feed Back* : idem, mais non parallélisable.
  - CFB, *Cipher Feed Back*, mode chaîné.

## Le mode dictionnaire ECB



- le clair doit être paddé
- Chaque bloc est chiffré indépendamment des autres blocs.

Figure – Dictionnaire ECB

# ECB en équations

## Chiffrement

$$C_1 = E_K(P_1)$$

$$C_2 = E_K(P_2)$$

...

$$C_\ell = E_K(P_\ell)$$

## Déchiffrement

$$P_1 = E_K^{-1}(C_1)$$

$$P_2 = E_K^{-1}(C_2)$$

...

$$P_\ell = E_K^{-1}(C_\ell)$$

## Cipher Bloc Chaining

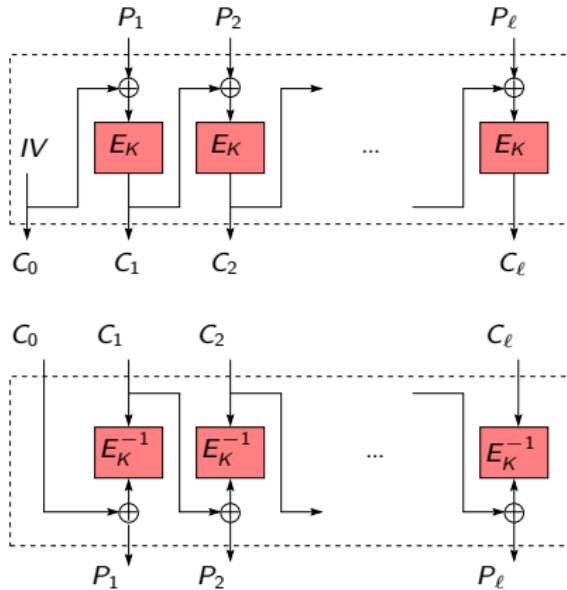


Figure – chaîne CBC

## CBC en équations

### Chiffrement

$$C_0 = IV \text{ aléatoire}$$

$$C_1 = E_K(P_1 \oplus IV)$$

$$C_2 = E_K(P_2 \oplus C_1)$$

...

$$C_\ell = E_K(P_\ell \oplus C_{\ell-1})$$

### Déchiffrement

$$P_1 = E_K^{-1}(C_1) \oplus C_0$$

$$P_2 = E_K^{-1}(C_2) \oplus C_1$$

...

$$P_\ell = E_K^{-1}(C_\ell) \oplus C_{\ell-1}$$

## Counter Mode

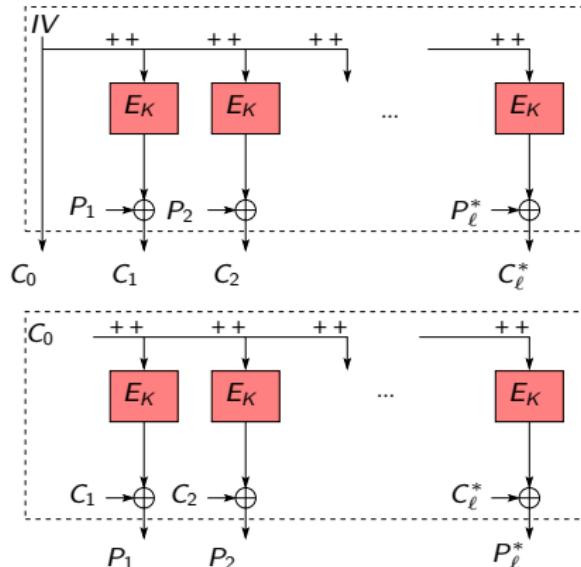


Figure – Mode compteur

- Pas de padding
- La variable auxiliaire  $/V$  est choisie aléatoirement. Elle est transmise comme “premier” bloc du chiffré.
- Elle est incrémentée la long de la chaîne par *incrémentation arithmétique*
- Le mode résulte en un chiffrement par flot (i.e. par addition de pseudo aléa)

## CTR en équations

### Chiffrement

$$C_0 = IV \text{ aléatoire}$$

$$C_1 = P_1 \oplus E_K(IV + 1)$$

$$C_2 = P_2 \oplus E_K(IV + 2)$$

...

$$C_\ell^* = P_\ell^* \oplus [E_K(IV + \ell)]^*$$

### Déchiffrement

$$P_1 = C_1 \oplus E_K(C_0 + 1)$$

$$P_2 = C_2 \oplus E_K(C_0 + 2)$$

...

$$P_\ell^* = C_\ell^* \oplus [E_K(C_0 + \ell)]^*$$

## Output Feed Back

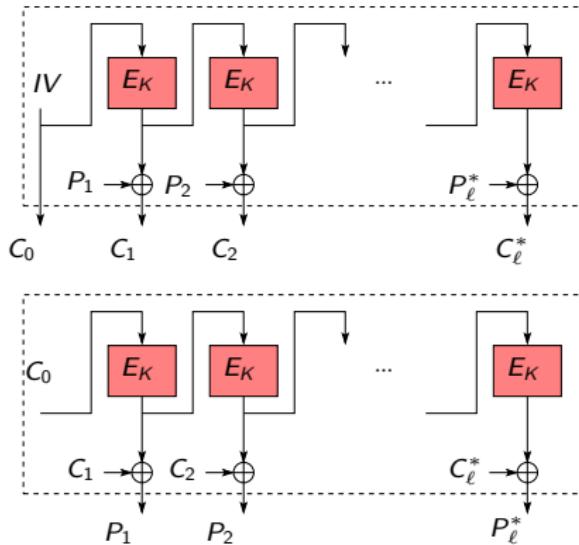


Figure – chaîne OFB

- Pas de padding
- La variable auxiliaire  $IV$  est choisie aléatoirement. Elle est transmise comme “premier” bloc du chiffré.
- Un registre est initialisé par  $IV$
- L'appel de la permutation sur la valeur courante donne une valeur aléatoire  $y$  qui est un bloc de pseudo aléa ;

# OFB en équations

## Chiffrement

$$C_0 = IV \text{ aléatoire}$$

$$Z_1 = E_K(IV), \quad C_1 = P_1 \oplus Z_1$$

$$Z_2 = E_K(Z_1), \quad C_2 = P_2 \oplus Z_2$$

...

$$Z_\ell = E_K(Z_{\ell-1}), \quad C_\ell^* = P_\ell^* \oplus [Z_\ell]^*$$

## Déchiffrement

$$Z_1 = E_K(C_0), \quad P_1 = C_1 \oplus Z_1$$

$$Z_2 = E_K(y_1), \quad P_2 = C_2 \oplus Z_2$$

...

$$Z_\ell = E_K(Z_{\ell-1}), \quad P_\ell^* = C_\ell^* \oplus Z_\ell^*$$

## Cipher Feed Back

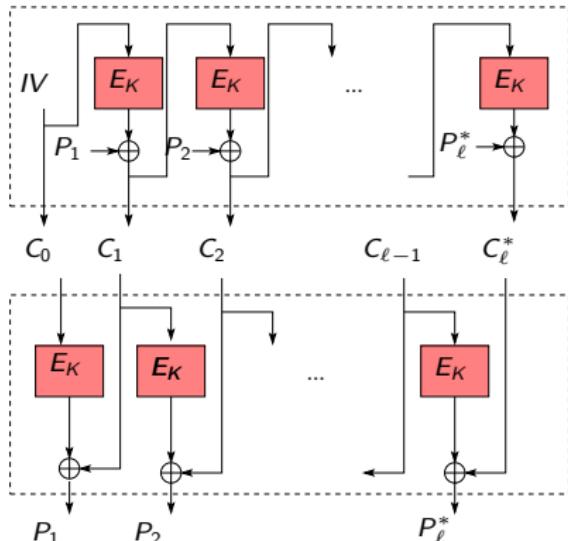


Figure – chaîne CFB

- Pas de padding
- La variable auxiliaire  $IV$  est choisie aléatoirement. Elle est transmise comme “premier” bloc du chiffré.
- Chaque bloc de chiffré sert d’entrée suivante de la permutation ;
- $IV$  est transmis comme “premier” bloc de chiffré

# CFB en équations

## Chiffrement

$$C_0 = IV \text{ aléatoire}$$

$$C_1 = P_1 \oplus E_K(IV)$$

$$C_2 = P_2 \oplus E_K(C_1)$$

...

$$C_\ell^* = P_\ell^* \oplus [E_K(C_{\ell-1})]^*$$

## Déchiffrement

$$P_1 = C_1 \oplus E_K(C_0)$$

$$P_2 = C_2 \oplus E_K(C_1)$$

...

$$P_\ell^* = C_\ell^* \oplus [E_K(C_{\ell-1})]^*$$

## Le *padding*

- Le padding est prévu par le concepteur comme partie intégrante du mode de chiffrement
- il ne doit pas être laissé à l'initiative de l'utilisateur non cryptologue
- ce dernier n'a donc pas à s'en soucier
- Exigence (cruciale pour la sécurité) : il doit s'appliquer à TOUS les messages.  
y compris – paradoxalement – si  $|P|$  est déjà multiple de  $b$ .  $P_{\text{pad}}$  contient alors un bloc entier de plus que  $P$ .

## Propriétés pratiques

- le padding et le dépadding doivent être faciles et rapides à mettre en œuvre  
leur calcul doit être marginal par rapport à celui du chiffrement/déchiffrement
- le dépadding doit être non ambigu  $\Rightarrow$  le padding doit être injectif

$$P \neq P' \Rightarrow P_{\text{pad}} \neq P'_{\text{pad}}$$

# Exemple

## Padding

- ① Écrire un “1” à la fin du message ;
- ② compléter par d’éventuels “0” jusqu’au prochain multiple de la taille du bloc

Illustration en exemple jouet :  $b = 8$

10100110		10001...		↪	10100110		10001	100	
10100110		1000....		↪	10100110		1000	1000	
10100110		1000110.		↪	10100110		1000110	1	
10100110		10001101		↪	10100110		10001101		10000000

## Dépadding

- ① Partir de la fin du message, barrer tous les éventuels “0” rencontrés
- ② Barrer le premier “1” rencontré

# Chiffrement symétrique

1 Chiffrement par flot

2 Chiffrement par bloc

3 Sécurité d'un chiffrement symétrique

- Niveau de sécurité
- Rappel : sécurité sémantique
- Indistinguabilité
- Illustrations

# Chiffrement symétrique

## 3 Sécurité d'un chiffrement symétrique

- Niveau de sécurité
- Rappel : sécurité sémantique
- Indistinguabilité
- Illustrations

## Niveau de sécurité / recouvrement de clé

Il est directement lié à recherche exhaustive sur la clé, donc à la taille de celle-ci :  $2^{|clé|}$ . Autrement dit

- la meilleure attaque est la recherche exhaustive sur la clé

Avec pour déclinaison :

- tout choix de clé est possible dans l'espace  $\{0, 1\}^{|clé|}$
- la clé doit être choisie aléatoirement dans cet espace

# Chiffrement symétrique

## 3 Sécurité d'un chiffrement symétrique

- Niveau de sécurité
- **Rappel : sécurité sémantique**
- Indistinguabilité
- Illustrations

# Sécurité sémantique d'un chiffrement symétrique. Approche traditionnelle/indistinguabilité

Objectifs :

- ne pas pouvoir retrouver la clé à clair connu
- ne pas pouvoir retrouver le clair à chiffré connu
- ne pas pouvoir deviner d'information partielle
- etc.

On va regrouper/concentrer tous ces objectifs en un seul, une condition suffisante, dite propriété maîtresse : *l'indistinguabilité left-right*

# Chiffrement symétrique

## 3 Sécurité d'un chiffrement symétrique

- Niveau de sécurité
- Rappel : sécurité sémantique
- **Indistinguabilité**
- Illustrations

## Indistinguabilité “left-right”

### Oracle de chiffrement “left-right”

- deux positions “gauche” et “droite” (parfois codée par un bit : 0=left, 1=right)
- un sélecteur de position
- deux entrées : clair de gauche, clair de droite
- une sortie : le chiffré
- agit comme suit : si le sélecteur est à gauche (resp. droite) chiffre le clair de gauche (resp. droite), ignore celui de droite (resp. gauche), et retourne le chiffré

## Le jeu IND-CPA (*Indistinguishability under Chosen Plaintext Attack*)

Le maître du jeu choisit la position *pos* du sélecteur

- au début du jeu et pour tout le jeu
- au hasard et honnêtement (*i.e.* avec probabilité 0.5/0.5)

L'attaquant *A*

- est à clair choisi (accès à un oracle classique de chiffrement)
- ne voit jamais le sélecteur
- interroge l'oracle *left-right* avec des couples de clairs de même longueur, reçoit un chiffré

$$A : \begin{array}{ccc} (M_{\text{left}}, M_{\text{right}}) & \longrightarrow & \mathcal{O} \\ \mathcal{E}_K(M_{\text{pos}}) & \longleftarrow & \end{array}$$

- doit deviner *pos*

## Avantage de l'attaquant

- Définition

$$\begin{aligned}\mathbf{Adv}^{rr}(A) &= 2 \left( \mathbb{P}(A \text{ devine bien}) - \frac{1}{2} \right) \\ &= \mathbb{P}(A \text{ retourne } \text{right} \mid \text{La position est right}) \\ &\quad - \mathbb{P}(A \text{ retourne } \text{right} \mid \text{La position est left})\end{aligned}$$

- Il mesure l'efficacité de  $A$
- s'exprime en fonction des paramètres de l'algorithme de chiffrement et du nombre de requêtes ;
- s'il est  $\leq 2^{-20}$ , l'algorithme de chiffrement est dit IND-CPA

## Synthèse

Sécurité	Attaquant	Proba	Avantage
Parfaite	Au hasard	$1/2$	0
Oui	Inefficace	$ p - 1/2  \leq 2^{-21}$	valeur absolue $\leq 2^{-20}$
Non	Efficace	$ p - 1/2  > 2^{-21}$	valeur absolue $> 2^{-20}$

## Le jeu IND-CCA

C'est presque le même jeu :

- contexte d'attaque à chiffré choisi :  
l'adversaire accède en plus à un oracle de déchiffrement  
→ plus favorable
- défi identique à celui de IND-CPA

En conséquence

- c'est un jeu plus facile à jouer pour l'attaquant
- IND-CCA sûr  $\Rightarrow$  IND-CPA sûr

# Chiffrement symétrique

## 3 Sécurité d'un chiffrement symétrique

- Niveau de sécurité
- Rappel : sécurité sémantique
- Indistinguabilité
- Illustrations

# Résultats

## Modes non sûrs

- ECB n'est pas IND-CPA
- Tout mode déterministe n'est pas IND-CPA
- avantage égal à 1 avec  $q = 2$  requêtes

## Les modes "sûrs"

- CTR, CBC, OFB CFB (randomisés) ont un avantage  $O(q^2/2^b)$ ,  $q$  : nombre de requêtes,  $b$  : taille bloc.
- ils sont IND-CPA tant que l'on ne traite pas plus de  $O(2^{b/2-10})$  blocs avec la même clé
- aucun n'est IND-CCA (avantage proche de 1 avec  $q = 2$  requêtes)

## ECB non sûr : démonstration visuelle

- Si plusieurs blocs d'un message sont identiques alors leurs valeurs de chiffrés seront également identiques.
- **Exemple.** Chiffrement ECB d'une photo où chaque pixel est un codé sur un bloc (128 bits)



Figure – Photo claire

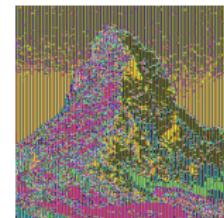


Figure – Photo chiffrée

## Exemple : distinguateur efficace sur le mode ECB

- 
- ① L'adversaire choisit deux blocs différents  $B_1$  et  $B_2$ .
  - ② Requête : avec deux messages de deux blocs chacun :
    - message gauche ( $B_1, B_1$ )
    - message droite ( $B_1, B_2$ ) $(C_1, C_2) :=$  réponse de l'oracle.
  - ③ si  $C_1 = C_2$  l'adversaire devine "gauche"
  - ④ si  $C_1 \neq C_2$ , l'adversaire devine "droite"

---

La probabilité de bien deviner est égale à 1, et donc l'avantage de cet adversaire est égal à 1.

# Application d'un mode sûr au chiffrement de la montagne

Avec un mode sûr, cela donne :



Figure – Photo claire

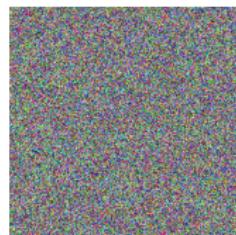


Figure – Photo chiffrée

– Partie 4 –  
Chiffrement asymétrique

# Chiffrement asymétrique

## 1 Principe

## 2 Problèmes mathématiques et algorithmes

## 3 Sécurité d'un chiffrement asymétrique

## 4 Autour du chiffrement asymétrique

# Chiffrement asymétrique

## Réflexions préliminaires

- c'est le clair qui est sensible
  - passer du chiffré au clair doit être difficile
  - passer du clair au chiffré peut être facile
- la clé et le procédé de chiffrement n'ont aucune raison d'être les mêmes que ceux de déchiffrement.

## Ideés sous-jacentes.

- Tout le monde peut chiffrer, une seule personne peut déchiffrer.
- pas de nécessité d'un secret commun préalable

## Mise en œuvre

Chaque utilisateur  $U$  possède une clé  $K$  se décomposant en deux parties :

- une partie publique notée  $K_{pu,U}$  et appelée *clé publique*, connue de tous (disponible par exemple dans un annuaire).
- une partie devant rester secrète notée  $K_{pr,U}$  et appelée *clé privée*.

Pour le chiffrement-déchiffrement :

- tout message clair  $P$ , le chiffrement  $E_K(P)$  se calcule avec la seule connaissance de la clé publique  $K_{pu}$  du destinataire
- pour tout message chiffré  $C$ , le déchiffrement  $E_K^{-1}(C)$  nécessite la connaissance de la partie privée  $K_{pr}$ .

$E_K : P \mapsto C$  est dite à sens unique à trappe

- Étant donné  $P$ , il est facile de calculer  $E_K(P)$
- Étant donné  $C$ , il est difficile de calculer  $E_K^{-1}(C) \dots$
- ...à moins de connaître une information supplémentaire : la *trappe*

## Illustration : cadenas et clés (1)

Les personnages sont Alice , et Bob .

- Alice veut envoyer un message à Bob, de manière sécurisée.
- Alice doit préalablement disposer d'un **cadenas ouvert** .
- Bob conserve la **clé** , correspondant au cadenas.



Le cadenas ouvert  est la **clé publique de Bob**. La clé  est la **clé privée de Bob**.

Les clés d'Alice ne sont pas utilisées !

## Illustration : cadenas et clés (2)

Mécanisme de chiffrement asymétrique :

- Alice place le message dans un coffre, fermé par le cadenas,
- Bob déverrouille le cadenas, et obtient le message.



Confidentialité : seul Bob peut lire le message d'Alice.

## Illustration : cadenas et clés (2)

Mécanisme de chiffrement asymétrique :

- Alice place le message dans un coffre, fermé par le cadenas,
- Bob déverrouille le cadenas, et obtient le message.



Confidentialité : seul Bob peut lire le message d'Alice.

## Illustration : cadenas et clés (2)

Mécanisme de chiffrement asymétrique :

- Alice place le message dans un coffre, fermé par le cadenas,
- Bob déverrouille le cadenas, et obtient le message.



Confidentialité : seul Bob peut lire le message d'Alice.

## Illustration : cadenas et clés (2)

Mécanisme de chiffrement asymétrique :

- Alice place le message dans un coffre, fermé par le cadenas,
- Bob déverrouille le cadenas, et obtient le message.



Confidentialité : seul Bob peut lire le message d'Alice.

## Illustration : cadenas et clés (2)

Mécanisme de chiffrement asymétrique :

- Alice place le message dans un coffre, fermé par le cadenas,
- Bob déverrouille le cadenas, et obtient le message.



Confidentialité : seul Bob peut lire le message d'Alice.

## Exigence fondamentale concernant les clés

Les clés publique et privée doivent être cohérentes avec le fait les fonctions  $E_K$  (resp.  $E_K^{-1}$ ) de chiffrement (resp. de déchiffrement) sont réciproques l'une de l'autre.

- Dans la spécification d'un algorithme de chiffrement asymétrique, la première phase (c'est nouveau) est la construction (une fois pour toutes) du couple (clé publique, clé privée).
- Il y a une relation mathématique entre les deux clés, que l'on peut même écrire algébriquement très simplement. En **théorie** la clé privée peut donc se déduire de la clé publique.
- La solidité du système repose sur la **difficulté calculatoire** de déduire la clé privée de la clé publique.
- Cette "difficulté calculatoire" est impliquée la **difficulté calculatoire d'un problème mathématique sous-jacent**.

# Chiffrement asymétrique

## 1 Principe

## 2 Problèmes mathématiques et algorithmes

- Factorisation des entiers et RSA
- La construction OAEP
- Logarithme discret et Elgamal
- Logarithme discret sur courbe elliptique

## 3 Sécurité d'un chiffrement asymétrique

## 4 Autour du chiffrement asymétrique

# Problèmes mathématiques et exemples de cryptosystèmes reliés

Les problèmes mathématiques dits difficiles viennent de domaines qui n'ont *a priori* rien à voir avec la cryptographie : théorie arithmétique des nombres, géométrie algébrique, théorie des codes correcteurs d'erreurs, arithmétique des polynômes, ...

## Exemples

- la *factorisation des entiers* : chiffrement RSA ;
- le *logarithme discret* : chiffrement Elgamal ;

Leur caractéristique essentielle pour leur intérêt cryptographique est qu'ils sont le problème inverse d'un problème dont la complexité est significativement plus petite :

- multiplication de deux entiers / factorisation

# Chiffrement asymétrique

## 2 Problèmes mathématiques et algorithmes

- Factorisation des entiers et RSA
- La construction OAEP
- Logarithme discret et Elgamal
- Logarithme discret sur courbe elliptique

# Factorisation des entiers

- **Problème.**

Connaissant  $n = pq$  produit de deux nombres premiers, trouver les valeurs de  $p$  et  $q$ .

- **Complexité.**

Produit de deux entiers :  $O(\log^2 n)$

Factorisation : *sous-exponentielle* en  $\log n$  :

$$O\left(e^{1.92(\log n)^{1/3}(\log \log n)^{2/3}}\right), \gg \text{poly}(\log n)$$

- **Record** (18 fév. 2020, nombre “non spécial”)

Entier  $n = \text{RSA-250}$ , entier de 829 bits.

- **Recommandation** (référentiel de l'ANSSI)

Utiliser des  $n$  de 3072 bits, avec  $p$  et  $q$  de taille 1536 bits.

# Chiffrement RSA (*Rivest, Shamir, Adleman*)

## Génération des clés

- génération de deux nombres premiers  $p$  et  $q$
- calcul de  $n = pq$
- choix d'un entier  $e$  premier avec  $\phi(n) = (p - 1)(q - 1)$
- calcul de  $d$  inverse de  $e$  modulo  $\phi(n)$ .  
Ainsi  $ed \equiv 1 \pmod{\phi(n)}$
- Clé publique :  $(n, e)$ . Clé privée :  $(p, q, d)$

## Critères

- $p$  et  $q$  1536 bits, et leur produit 3072 :  $2^{1535.5} \leq p, q \leq 2^{1536}$
- $e$  et  $d$  de même ordre de grandeur que  $n$ .

Chiffrement :

$$E_K : \begin{array}{ccc} (\mathbb{Z}/n\mathbb{Z})^\times & \rightarrow & (\mathbb{Z}/n\mathbb{Z})^\times \\ m & \mapsto & c = m^e \bmod n \end{array}$$

Déchiffrement :

$$E_K^{-1} : \begin{array}{ccc} (\mathbb{Z}/n\mathbb{Z})^\times & \rightarrow & (\mathbb{Z}/n\mathbb{Z})^\times \\ c & \mapsto & m = c^d \bmod n \end{array}$$

# Chiffrement RSA : pourquoi ça marche ?

## Consistance des calculs RSA

$$\begin{aligned}E_K^{-1}(E_K(m)) &= (E_K(m))^d \bmod n \\&= m^{ed} \bmod n \\&= m^{1+\alpha\phi(n)} \bmod n \text{ (pour un certain } \alpha \in \mathbb{N}) \\&= m(m^{\phi(n)})^\alpha \bmod n \\&= m 1^\alpha \bmod n \\&= m \bmod n\end{aligned}$$

## Exemple numérique

Clés :

- $p = 13, q = 19$
- $n = 13 \times 19 = 247$
- $\phi(n) = (13 - 1) \times (19 - 1) = 216$
- $e = 23, d = 47$  car  $23 \times 47 \equiv 1 \pmod{216}$

Chiffrement :

$$m = 7, \text{ alors } c = 7^{23} \pmod{247} = 106$$

Déchiffrement :

$$106^{47} = 7 \pmod{247}$$

# Sécurité du chiffrement RSA

Recouvrement de clé (privée). Sont équivalentes :

- la connaissance de la factorisation  $n = pq$
- la connaissance de  $\phi(n)$
- la connaissance de l'exposant privé  $d$

Sécurité sémantique

- problèmes de la primitive RSA
  - déterministe, donc distinguable left-right
  - propriété multiplicative :  $E_K(m_1m_2) = E_K(m_1) E_K(m_2)$
- solution : appliquer une encapsulation de type **OAEP** ou PKCS11 incluant formatage, padding et randomisation.

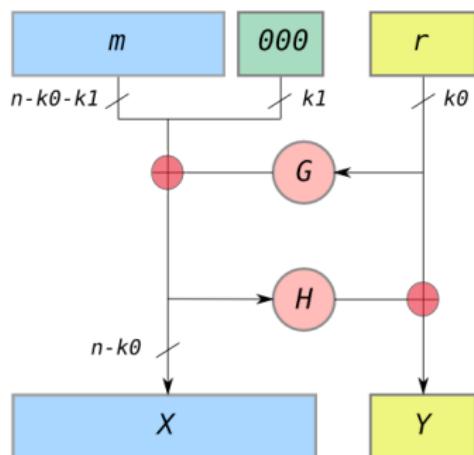
# Chiffrement asymétrique

## 2 Problèmes mathématiques et algorithmes

- Factorisation des entiers et RSA
- **La construction OAEP**
- Logarithme discret et Elgamal
- Logarithme discret sur courbe elliptique

# Le formatage OAEP

*Optimal Asymmetric Encryption Padding*



- $G : \{0, 1\}^{k_0} \longrightarrow \{0, 1\}^{n-k_0}$ ,
- $H : \{0, 1\}^{n-k_0} \longrightarrow \{0, 1\}^{k_0}$ .
- Random  $r$  : randomisation ( $k_0 \simeq 256$ ),
- Padding  $0^{k_1}$  : contrôle de validité ( $k_1 \simeq 128$ ).

# Le chiffrement RSA-OAEP

Chiffrement :

$$\begin{array}{ccccccc} \{0,1\}^{n-k_0-k_1} & \rightarrow & \{0,1\}^n & \subset (\mathbb{Z}/N\mathbb{Z})^\times & \rightarrow & (\mathbb{Z}/N\mathbb{Z})^\times \\ m & \xrightarrow{\text{OAEP}} & (X, Y) : \left\{ \begin{array}{lcl} X & = & (m \parallel 0) \oplus G(r) \\ Y & = & H(X) \oplus r \end{array} \right. & & \xrightarrow{\text{RSA}} & C \end{array}$$

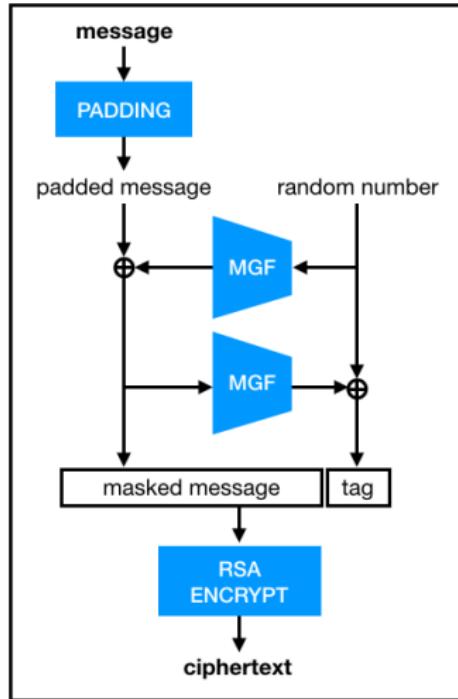
Déchiffrement :

$$\begin{array}{ccccccc} (\mathbb{Z}/N\mathbb{Z})^\times & \rightarrow & \{0,1\}^n & \rightarrow & \{0,1\}^{n-k_0} \times \{0,1\}^{k_0} \\ C & \xrightarrow{\text{RSA}^{-1}} & (X, Y) & \xrightarrow{\text{OAEP}^{-1}} & (r, w) : \left\{ \begin{array}{lcl} r & = & Y \oplus H(X) \\ w & = & X \oplus G(r) \end{array} \right. \end{array}$$

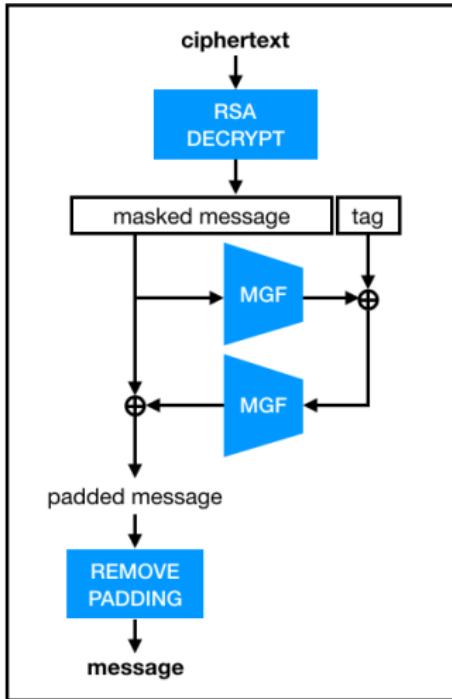
- si  $w$  est de la forme  $(m \parallel 0^{k_1})$ , retourner  $m$
- sinon retourner "chiffré invalide"

Sécurité :

- IND-CCA
- *Plaintext awareness* (conscience du clair) : assurance que le chiffré a été calculé par la procédure prévue.



encryption with RSA-OAEP



decryption with RSA-OAEP

# Chiffrement asymétrique

## 2 Problèmes mathématiques et algorithmes

- Factorisation des entiers et RSA
- La construction OAEP
- **Logarithme discret et Elgamal**
- Logarithme discret sur courbe elliptique

# Logarithme discret

## Problème

Soit un  $(\mathbb{G}, \cdot)$  un groupe cyclique et  $g$  un générateur. Soit  $y \in \mathbb{G}$ . Trouver  $\ell \in \mathbb{N}$  tel que  $y = g^\ell$ .

C'est l'opération inverse de l'exponentiation dans le groupe  $G$ .

- si  $\ell$  convient, alors tout  $\ell' \equiv \ell \pmod{\#\mathbb{G}}$  convient aussi. Il suffit donc de chercher  $\ell$  dans  $\{0, \dots, \#\mathbb{G} - 1\}$
- $\ell$  est unique si on impose  $\ell \in \{0, \dots, \#\mathbb{G} - 1\}$
- on note  $\ell = \log_g y$

En cryptographie, on s'intéresse aux groupes  $\mathbb{G}$  où l'exponentiation  $\ell \mapsto g^\ell$  est significativement plus rapide que le calcul de logarithme discret.

## Logarithme discret dans $\mathbb{F}_p^*$

- **Problème DL** : pour  $p$  premier,  $g, h \in \mathbb{F}_p^*$ ,  $g$  générateur, trouver  $\ell$  (entier) tel que  $h = g^\ell \text{ mod } p$ .
  - $\ell$  est unique si restreint à  $\{0, \dots, p-2\}$
  - appelé le *logarithme discret modulaire* de  $h$  en base  $g \text{ mod } p$ .
- **Complexité.**
  - Exponentiation :  $O(\log p)$
  - Logarithme discret :  $O\left(e^{1.92(\log p)^{1/3}(\log \log p)^{2/3}}\right)$
- **Recommandation.** Choisir  $p$  :
  - au moins 3072 bits
  - $p - 1$  a au moins un facteur premier de 256 bits

P. ex.,  $p$  est un nombre premier *sûr*, càd  $p = 2q + 1$  avec  $q$  premier ;  $q$  s'appelle un nombre premier de Sophie Germain

# Chiffrement Elgamal dans $\mathbb{F}_p^*$

## Génération des clés

- génération un nombre premier  $p$ ,
- choix d'un générateur  $g$  du groupe multiplicatif  $\mathbb{F}_p^*$ ,
- choix d'un entier  $K_{pr} \in \{1, \dots, p - 2\}$
- calcul de  $K_{pu} = h = g^{K_{pr}} \bmod p$
- Clé publique :  $(p, g, h)$ . Clé privée :  $K_{pr}$ .

Chiffrement :  $\mathbb{F}_p^* \rightarrow \mathbb{F}_p^* \times \mathbb{F}_p^*$ ,  $m \mapsto c = (c_1, c_2)$

- Choisir  $r$  aléatoire **secret**  $\in \{1, \dots, p - 2\}$
- $(c_1, c_2) = (\textcolor{red}{g}^r \bmod p, m\textcolor{red}{h}^r \bmod p)$

Déchiffrement :  $\mathbb{F}_p^* \times \mathbb{F}_p^* \rightarrow \mathbb{F}_p^*$ ,  $c = (c_1, c_2) \mapsto m$

- $m = c_2 \left( c_1^{\textcolor{red}{K}_{pr}} \right)^{-1} \bmod p$

NB : formule équivalente (mieux pour application numérique) :  $m = c_2 c_1^{\textcolor{red}{p}-1 - K_{pr}} \bmod p$

## Elgamal : pourquoi ça marche

Calcul algébrique direct, sans astuce ni théorème particulier

$$\begin{aligned} E_K^{-1}(E_K(m)) &= [m \cdot h^r] \cdot [(g^r)^{K_{pr}}]^{-1} \\ &= [m \cdot (g^{K_{pr}})^r] \cdot [(g^r)^{K_{pr}}]^{-1} \\ &= m \cdot g^{K_{pr}r - rK_{pr}} \\ &= m \cdot 1 \\ &= m \end{aligned}$$

# Exemple numérique

## Clés

- $p = 101$ ;  $g = 2$  est générateur de  $\mathbb{F}_{101}^*$

$$\mathbb{F}_{101}^* = \{2^0, 2^1, \dots, 2^{99}\} = \{1, 2, \dots, 100\}$$

- $\ell = 37$ , et alors  $h = 2^{37} \bmod 101 = 55$

## Chiffrement

$m = 18$

- (avec  $r = 11$ ),  $(c_1, c_2) = (2^{11}, 18 \times 55^{11}) = (28, 82) \bmod 101$
- (avec  $r = 13$ ),  $(c_1, c_2) = (2^{13}, 18 \times 55^{13}) = (11, 95) \bmod 101$

## Déchiffrement

- 1er cas :  $(82 \times (28^{37})^{-1}) = 18 \bmod 101$
- 2me cas :  $(95 \times (11^{37})^{-1}) = 18 \bmod 101$

# Chiffrement asymétrique

## 2 Problèmes mathématiques et algorithmes

- Factorisation des entiers et RSA
- La construction OAEP
- Logarithme discret et Elgamal
- Logarithme discret sur courbe elliptique

## Le cas $\mathbb{G} = \mathbb{E}$ , courbe elliptique

### Courbes elliptiques en contexte cryptographique

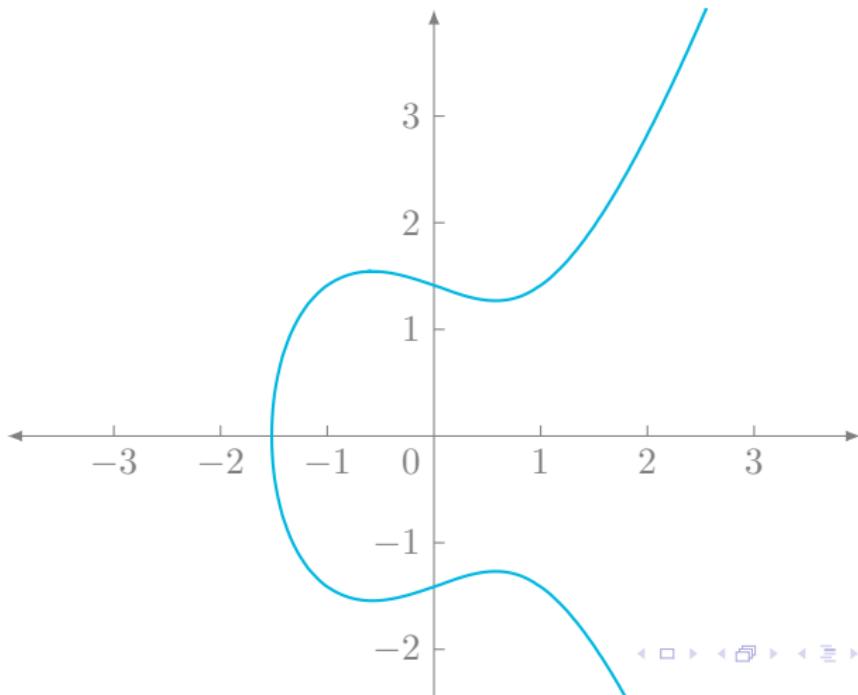
- Soit  $p$  un nombre premier  $\neq 2, 3$ , et  $a, b \in \mathbb{F}_p$ . Une courbe elliptique est l'ensemble

$$\mathbb{E}(\mathbb{F}_p) = \{O_{\mathbb{E}}\} \cup \{(X, Y) \in \mathbb{F}_p^2, Y^2 = X^3 + aX + b\}.$$

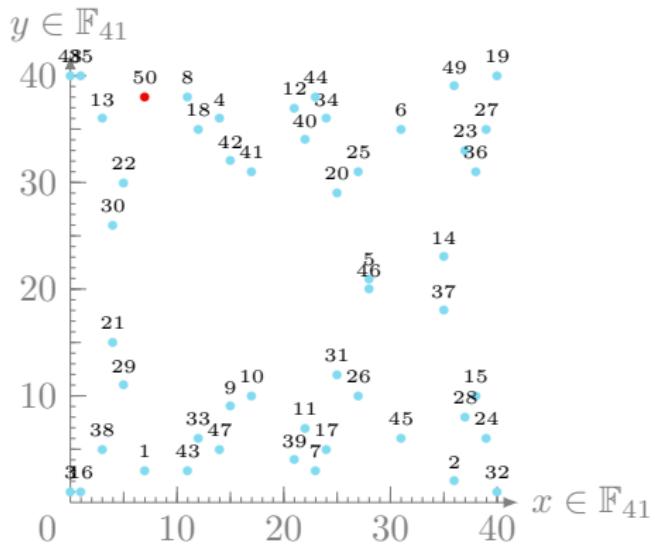
où  $O_{\mathbb{E}}$  est un point spécial dit “point à l'infini”.

- $\mathbb{E}(\mathbb{F}_p)$  est un ensemble fini
  - ATTENTION** : *a priori*  $\#\mathbb{E}(\mathbb{F}_p) \neq p$
  - borne de Weil  $\#\mathbb{E}(\mathbb{F}_p) \in [\lfloor p + 1 - 2\sqrt{p} \rfloor; \lceil p + 1 + 2\sqrt{p} \rceil]$
  - donc  $\#\mathbb{E}(\mathbb{F}_p) \sim p$  quand  $p \rightarrow +\infty$

## Représentation d'une courbe elliptique sur $\mathbb{R}$

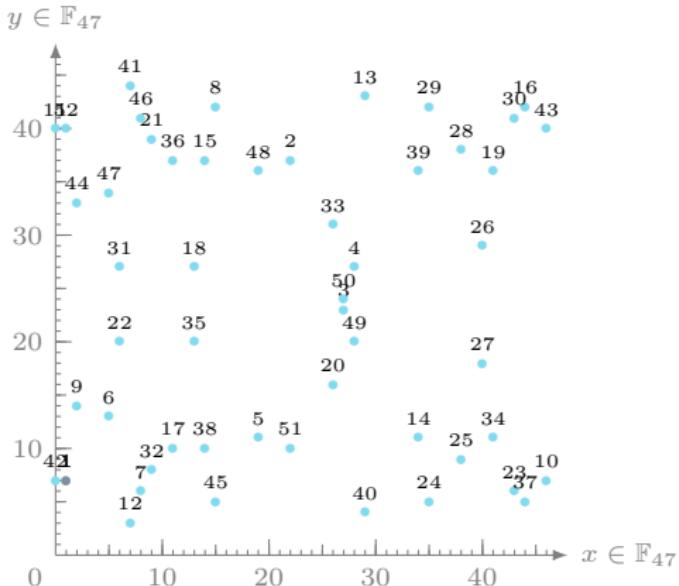


# Représentation d'une courbe elliptique sur un corps fini



$\{y^2 = x^3 - x + 1\}$  sur  $\mathbb{F}_{41}$ . Cardinal : 50

# Représentation d'une courbe elliptique sur un corps fini



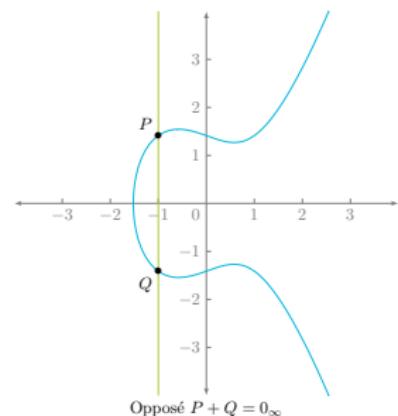
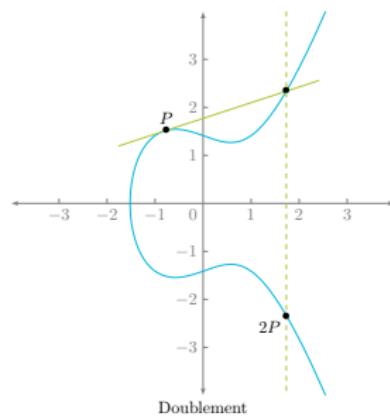
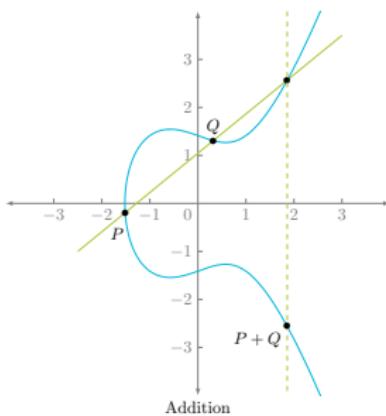
$$y^2 = x^3 - x + 2 \text{ sur } \mathbb{F}_{47}. \text{ Cardinal : 53}$$

## Loi de composition interne

Notée “+”.

- $P + O_{\mathbb{E}} = O_{\mathbb{E}} + P = P$ , par convention.  
 $\Rightarrow O_{\mathbb{E}}$  l'élément neutre de +
- $P \neq Q$ , (“addition”) on trace la droite  $(PQ)$ 
  - si elle recoupe  $\mathbb{E}$  en  $R$ . Alors  $P + Q = \text{Sym}_{(Ox)}(R)$
  - sinon, elle est verticale, et  $P + Q = O_{\mathbb{E}}$   
 $\Rightarrow Q$  est le symétrique de  $P$  pour +, noté  $-P$
- $P = Q$ , (“doublement”) on trace la tangente en  $P$  à  $\mathbb{E}$ 
  - si  $P \notin (Ox)$ , elle recoupe  $\mathbb{E}$  en  $R$ . Alors  $P + P$ , noté  $2P$ ,  
 $= \text{Sym}_{Ox}(R)$
  - si  $P \in (Ox)$ , elle est verticale et  $P + P = O_{\mathbb{E}}$

## Avec un (joli) dessin (dans le cas réel)



## Avec les formules (dans tous les cas)

On pose  $P = (x_P, y_P)$ ,  $Q = (x_Q, y_Q)$ . Alors  
 $P + Q = (x_{P+Q}, y_{P+Q})$  est donné par

$$\begin{cases} x_{P+Q} &= s^2 - x_P - x_Q \\ y_{P+Q} &= -sx_{P+Q} - t \end{cases}$$

où  $y = sx + t$  est l'équation de  $(PQ)$  (resp. de la tangente en  $P$ ),  
càd

$$s = \begin{cases} \frac{y_Q - y_P}{x_Q - x_P} & \text{si } P \neq Q \\ \frac{3x_P^2 + a}{2y_P} & \text{si } P = Q \end{cases}, \quad t = y_P - sx_P$$

## Structure de groupe

$(\mathbb{E}(\mathbb{F}_p), +)$  est un groupe abélien

- $+$  est commutative :  $P + Q = Q + P$
- $+$  est associative :  $P + (Q + R) = (P + Q) + R$
- $+$  a un élément neutre :  $O_{\mathbb{E}}$
- tout point  $P(x_P, y_P)$  admet un symétrique noté  $-P(x_P, -y_P)$

De plus, si  $\#\mathbb{E}(\mathbb{F}_p)$  est un nombre premier,

- $(\mathbb{E}(\mathbb{F}_p), +)$  est un groupe cyclique
- tout point  $\neq O_{\mathbb{E}}$  est un générateur

# Le logarithme discret sur courbes elliptiques

*“Exponentielle” d'un point.*

Pour  $P \in \mathbb{E}(\mathbb{F}_p)$  et  $\ell \in \mathbb{N}$ , on note  $\ell P = [\ell]P = \underbrace{P + \cdots + P}_{\ell \text{ fois}}$ .

## Problème ECDL.

étant donné  $p$  nombre premier  $\neq 2, 3$ ,  $\mathbb{E}(\mathbb{F}_p)$  une courbe elliptique,  
 $P, Q \in \mathbb{E}(\mathbb{F}_p)$ , trouver  $\ell \in \mathbb{N}$  (s'il existe) satisfaisant  $Q = \ell P$ .

## Complexité.

Exponentielle d'un point :  $\text{poly}(\log p)$

Logarithme discret :  $O(\sqrt{p}) = \text{exponentielle en } \log p$

# Chiffrement Elgamal sur courbe elliptique

## Génération des clés

- générer un nombre premier  $p \neq 2, 3$
- choisir une courbe elliptique  $\mathbb{E}(\mathbb{F}_p)$  de cardinal premier, donc cyclique
- choisir  $P \neq O_{\mathbb{E}}$  de  $\mathbb{E}(\mathbb{F}_p)$  (c'est automatiquement un générateur)
- choisir un entier  $K_{pr} \in \{1, \dots, \#\mathbb{E}(\mathbb{F}_p) - 1\}$ ,
- calculer  $Q = K_{pr}P$
- Clé publique :  $(p, \mathbb{E}(\mathbb{F}_p), \#\mathbb{E}(\mathbb{F}_p), P, Q)$ . Clé privée :  $K_{pr}$ .

## Chiffrement

- Choisir aléatoirement un entier **secret**  
 $r \in \{1, \dots, \#\mathbb{E}(\mathbb{F}_p) - 1\}$
- $E_K(M) = (rP, M + rQ)$

## Déchiffrement

- $E_K^{-1}((C_1, C_2)) = C_2 - K_{pr}C_1$

## Sécurité en recouvrement de clé du chiffrement Elgamal

La sécurité repose entre autre sur la difficulté de trouver la clé privée connaissant la clé publique.

- Cas  $\mathbb{F}_p^*$  :  
connaissant  $p, g, h$ , trouver  $K_{pr}$  tel que  $g^{K_{pr}} = h \bmod p$
- Cas courbe elliptique : connaissant  $\mathbb{E}(\mathbb{F}_p), P, Q$ , trouver  $K_{pr}$  tel que  $[K_{pr}]P = Q$

Ce problème est exactement celui du logarithme discret :

- Cas  $\mathbb{F}_p^*$  : de  $h$  en base  $g$  modulo  $p$ .
- Cas courbe elliptique : de  $Q$  en base  $P$  dans  $\mathbb{E}(\mathbb{F}_p)$ .

## Comparatif RSA - DL $\mathbb{F}_p$ - DL courbes elliptiques

Théorique/intrinsèque :

- DL sur courbes elliptiques, pas de meilleur algorithme que les algorithmes génériques
- DL sur  $\mathbb{F}_p$ , il y a des attaques spécifiques

Pratique : pour un niveau de sécurité donné, tailles de clés plus petites sur courbes elliptiques

Sécurité	Taille du $n$ dans RSA . Taille du $p$ dans DL sur $\mathbb{F}_p$	Taille du $p$ pour ECDL
$2^{80}$	1 024	160
$2^{128}$	3 072	256
$2^{256}$	15 360	512

# Chiffrement asymétrique

## 1 Principe

## 2 Problèmes mathématiques et algorithmes

## 3 Sécurité d'un chiffrement asymétrique

- Niveau de sécurité
- Indistinguabilité

## 4 Autour du chiffrement asymétrique

# Chiffrement asymétrique

## 3 Sécurité d'un chiffrement asymétrique

- Niveau de sécurité
- Indistinguabilité

## Niveau de sécurité vs taille de clé

Contrairement au cas symétrique, le niveau de sécurité est **indirectement** relié à la taille de la clé.

- l'attaque générique n'est plus la brute force mais celle qui résout le problème mathématique sous-jacent ;
- or la complexité du problème math sous-jacent est «<< complexité brute force  $2^{|clé privée|}$  (ca c'est la vie...) »

# Chiffrement asymétrique

## 3 Sécurité d'un chiffrement asymétrique

- Niveau de sécurité
- Indistinguabilité

## Sécurité sémantique par indistinguabilité

Il s'agit du même jeu qu'en chiffrement symétrique.

- Un oracle “left-right”
- un maître du jeu choisit une position
- contexte clair choisi est natif pour IND-CPA.
- oracle de déchiffrement nécessaire pour IND-CCA.
- l'adversaire doit deviner la position
- son efficacité est mesurée par l'avantage :  
 $\mathbf{Adv} = 2(\mathbb{P}(\text{bien deviner}) - 1/2)$
- algorithme sûr si  $\mathbf{Adv} \leq 2^{-20}$
- IND-CCA sûr  $\Rightarrow$  IND-CPA sûr

## Sécurité par indistinguabilité

On introduit une propriété maîtresse : *l'indistinguabilité left-right*.  
L'oracle de chiffrement “*left-right*”

- a deux entrées : le clair de gauche et celui de droite
- a un sélecteur caché
- a une sortie : le chiffré
- agit comme suit : si le sélecteur est à gauche (resp. droite) chiffre le clair de gauche (resp. droite), ignore celui de droite (resp. gauche), et retourne le chiffré

## Le jeu IND-CPA

- Le maître du jeu choisit au hasard honnêtement la position du sélecteur au début du jeu et pour tout le jeu. On code par un bit  $b$  cette position : 0=left, 1=right.
- L'attaquant  $A$  interroge l'oracle avec des couples de clairs  $\underbrace{M_0}_{\text{left}}, \underbrace{M_1}_{\text{right}}$ . L'oracle retourne  $\mathcal{E}_K(M_b)$ . L'attaquant :
  - doit respecter  $|M_0| = |M_1|$ , sinon distinguer est évident et non pertinent.
  - peut prendre  $M_0 = M_1$  de sorte que l'oracle left-right peut être utilisé comme un oracle ordinaire de chiffrement. Le contexte est donc à clair choisi. **L'accès à l'oracle de chiffrement est en fait inutile car le chiffrement est public**
- L'attaquant doit finalement deviner  $b$ , càd la position du sélecteur choisie par le maître du jeu.

## Avantage

On mesure l'efficacité de  $A$  par son avantage

$$\begin{aligned}\mathbf{Adv}_{\mathcal{E}}^{lr}(A) &= \Pr(A \rightarrow 1 | Right) - \Pr(A \rightarrow 1 | Left) \\ &= 2\Pr(A \text{ devine bien}) - 1\end{aligned}$$

L'avantage :

- est calculé comme une fonction des paramètres du schéma de chiffrement et du nombre de requêtes ;
- s'il est  $\leq 2^{-20}$ , le schéma de chiffrement est dit IND-CPA

Sécurité	Attaquant	Proba $p$	Avantage
Parfaite	Au hasard	$1/2$	0
Oui	Inefficace	$ p - 1/2  \leq 2^{-21}$	valeur absolue $\leq 2^{-20}$
Non	Efficace	$ p - 1/2  > 2^{-21}$	valeur absolue $> 2^{-20}$

Si l'adversaire accède en plus à un oracle de déchiffrement (contexte à chiffré choisi), on définit un jeu analogue dit IND-CCA.

IND-CCA sûr  $\Rightarrow$  IND-CPA sûr

# Performances

## Le chiffrement asymétrique

- permet de se passer du secret commun préalable
- permet d'avoir moins de clés dans le système en cas de nombreux utilisateurs
- MAIS est 100 à 1000 fois plus lent que son équivalent symétrique

## Conséquences

- on limite le chiffrement asymétrique aux messages de petite taille

# Chiffrement asymétrique

## 1 Principe

## 2 Problèmes mathématiques et algorithmes

## 3 Sécurité d'un chiffrement asymétrique

## 4 Autour du chiffrement asymétrique

- Chiffrement hybride
- Négociation de clé Diffie-Hellman
- Certification de clé publique

# Chiffrement asymétrique

## ④ Autour du chiffrement asymétrique

- Chiffrement hybride
- Négociation de clé Diffie-Hellman
- Certification de clé publique

# Chiffrement hybride

## Objectif

- Chiffrer un message d'Alice vers Bob sans secret commun préalable

## Solution basique

- Utiliser un chiffrement asymétrique en définissant l'analogie de modes d'opération

Considérer deux faits :

- le message clair  $M$  peut être grande taille et impliquer le chiffrement de nombreux blocs
- le chiffrement asymétrique est beaucoup plus lent que le chiffrement symétrique

## Une solution hybride

- chiffrer le message en symétrique avec une clé secrète
- chiffrer la clé secrète en asymétrique

## Cumul d'avantages

- rapidité de chiffrement (grâce au symétrique)
- propriété agréable “pas de secret commun préalable” (grâce à l'asymétrique)

## Mise en œuvre

### Chiffrement hybride du message $M$

- ➊ Alice choisit une valeur secrète  $(tmp)_{secret\_key}$
- ➋ Alice calcule  $C_1 = \text{AsymEnc}_{Bob\_public\_key}((tmp)_{secret\_key})$  ;
- ➌ Alice calcule  $C_2 = \text{SymEnc}_{(tmp)_{secret\_key}}(M)$
- ➍ Alice envoie  $(C_1, C_2)$  à Bob.

### Déchiffrement de $(C_1, C_2)$

- ➊ Bob calcule  $\text{AsymEnc}_{Bob\_private\_key}^{-1}(C_1)$  et retrouve  $(tmp)_{secret\_key}$
- ➋ Bob calcule  $\text{SymEnc}_{(tmp)_{secret\_key}}^{-1}(C_2)$  et retrouve  $M$

## Comparatif

Chiffrement asymétrique pur de  $M$  :

- chiffrer “lentement” un “gros” message

Chiffrement hybride, au bilan plus efficace

- chiffrer “lentement” un “petit” message  $((tmp)_secret\_key)$ ,  
puis
- chiffrer “rapidement” un “gros” message ( $M$ )

# Chiffrement asymétrique

## 4 Autour du chiffrement asymétrique

- Chiffrement hybride
- Négociation de clé Diffie-Hellman
- Certification de clé publique

## Protocole de Diffie-Hellman

Il s'agit d'un mécanisme dit d'*échange de clé*, bien que le libellé *négociation de clé* ou “génération d'un secret commun” soit plus adapté. Alternative interactive à l'étape 1 du chiffrement hybride.

- Alice et Bob contribuent tous deux à la construction de leur (futur) secret commun
- aucun des deux n'en connaît à l'avance la valeur
- aucun des deux ne peut cibler une valeur donnée à l'avance

Données publiques :

- un groupe cyclique  $\mathbb{G}$  dans lequel le logarithme discret est difficile.
- un générateur  $g$  de  $\mathbb{G}$ .

## Protocole de Diffie-Hellman dans $\mathbb{G} = \mathbb{F}_p^*$ .

Alice	Bob
1 : $a \in \{1, \dots, p-2\}$ aléa	
2 :	$A := \underbrace{g^a}_{\longrightarrow} \bmod p$
3 :	
4 :	$b \in \{1, \dots, p-2\}$ aléa
	$B := \underbrace{g^b}_{\longleftarrow} \bmod p$
5 : $B^a \bmod p$	
6 :	$A^b \bmod p$

Secret commun :  $g^{ab} \bmod p$

## Sécurité passive

Il repose sur la difficulté du problème dit de Diffie-Hellman.

étant donnés :

- un groupe cyclique  $\mathbb{G}$  de générateur  $g$ ,
- deux éléments **secrets**  $a, b \in \{1, \dots, \#\mathbb{G} - 1\}$ ,

peut-on déterminer  $g^{ab}$  en ne connaissant que  $g^a$  et  $g^b$  (mais pas  $a$  ni  $b$ ) ?

Ce problème est lié à celui du logarithme discret dans  $\mathbb{G}$  : il est (strictement ?) moins difficile.

Le protocole de Diffie-Hellmann requiert que le canal soit intégrer. Sinon, il existe une attaque par le milieu.

## (In)sécurité active : attaque par le milieu

	Alice	Eve	Bob
1 :	$a$ aléa		
2 :		$\xrightarrow{g^a}$	
3 :		$a^*$ aléa	
4 :			$\xrightarrow{g^{a^*}}$
5 :			$b$ aléa
6 :			$\xleftarrow{g^b}$
7 :		$b^*$ aléa	
8 :		$\xleftarrow{g^{b^*}}$	
9 :	$(g^{b^*})^a$		
10 :			$(g^{a^*})^b$
11 :		$(g^a)^{b^*}$	
12 :		$(g^b)^{a^*}$	

## La parade : DH authentifié

Alice	Bob
1 : $a$ aléa	
2 : $g^a$ authentifié	→
3 :	$b$ aléa
4 : $g^b$ authentifié	←
5 : $(g^b)^a$	
6 : $(g^a)^b$	

L'authentification réfère ici à la preuve d'origine (*cf.* intégrité)

## Diffie-Hellman sur courbe elliptique (ECDH)

$P$  un point générateur d'une courbe  $\mathbb{E}(\mathbb{F}_p)$

	Alice	Bob
1 :	$a \in \{1, \dots, \#\mathbb{E} - 1\}$ aléa	
2 :		$aP$ authentifié $\xrightarrow{ }$
3 :		
4 :	$bP$ authentifié $\xleftarrow{ }$	$b \in \{1, \dots, \#\mathbb{E} - 1\}$ aléa
5 :	$a(bP)$	
6 :		$b(aP)$

Secret commun : le point  $abP$  de la courbe

# Chiffrement asymétrique

## 4 Autour du chiffrement asymétrique

- Chiffrement hybride
- Négociation de clé Diffie-Hellman
- Certification de clé publique

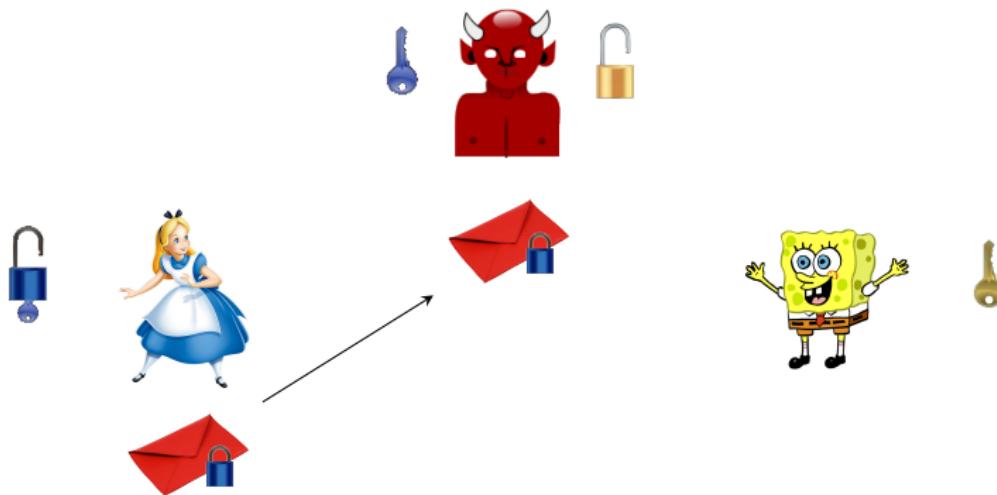
# Attaque d'une clé publique non certifiée



# Attaque d'une clé publique non certifiée



## Attaque d'une clé publique non certifiée



## Attaque d'une clé publique non certifiée



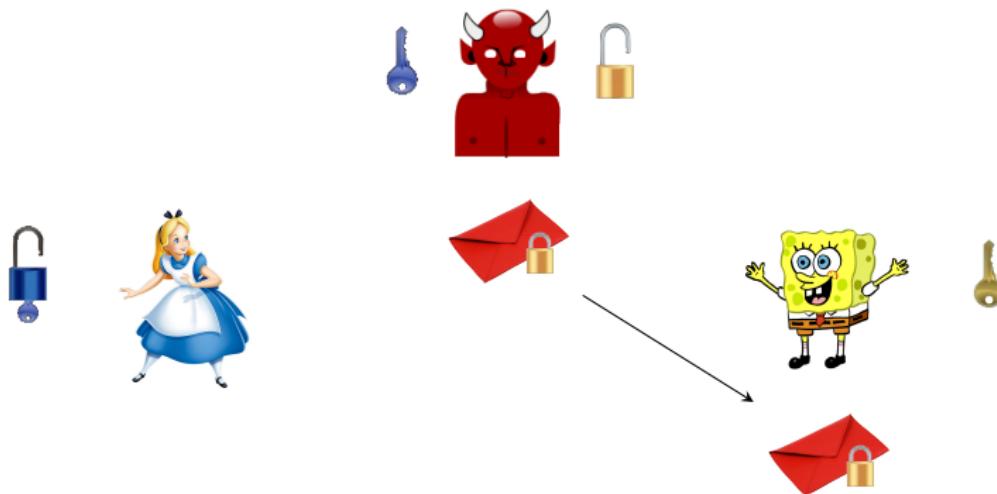
# Attaque d'une clé publique non certifiée



## Attaque d'une clé publique non certifiée



## Attaque d'une clé publique non certifiée



# Attaque d'une clé publique non certifiée



# Attaque d'une clé publique non certifiée



## La gestion des clés publiques

Les clés publiques ne sont pas confidentielles mais doivent rester **intègres**, ce qui implique une gestion cryptographique par des moyens que l'on verra par ailleurs (signature).

On ne peut donc pas éviter l'*infrastructure de gestion des clés*. Dans le cas de la cryptographie à clé publique, citons notamment comme acteur l'*autorité de certification*, tierce partie de confiance qui “assure” les clés publiques aux utilisateurs.

## – Partie 5 – Intégrité symétrique. Hachage

# Intégrité symétrique. Hachage

1 Intégrité symétrique. MAC

2 Sécurité d'un MAC

3 Fonction de hachage

4 Constructions de MAC

5 Intégrité et confidentialité combinées

# Intégrité : introduction

## Une vieille solution utilisant le chiffrement symétrique

- Alice chiffre le message  $M$  et obtient  $C$
- Un adversaire modifie en aveugle  $C$  en  $C^*$
- Bob déchiffre  $C^*$  et obtient  $M^*$

L'adversaire ne connaît pas la clé secrète. Quand il modifie le chiffré, il ne peut pas contrôler la modification induite sur le clair. Il est très improbable que  $M^*$  soit sémantiquement pertinent. Ainsi Bob détecte que le message a été modifié.

Ce raisonnement n'est valide que dans des contextes spécifiques.

## La bonne solution

Utiliser un *Message Authentication Code, MAC*, code d'authentification de message

## Définition d'un MAC

Un MAC est une fonction  $\text{MAC}_K$  indexée par la clé secrète  $K$  qui, étant donné un message de longueur quelconque  $M$ , calcule une valeur de longueur fixe  $T = \text{MAC}_K(M)$ , tel que le calcul soit :

- facile quand  $K$  est connue
- pratiquement impossible quand  $K$  est inconnue

Un MAC est l'analogue d'un CRC dans le domaine de la détection d'erreurs. Il doit être conditionné à une clé secrète pour détecter une modification, car celle-ci est mal intentionnée et ciblée et non pas accidentelle et aléatoire

## Schéma général d'une transmission intègre

Alice et Bob partagent un algorithme de MAC et une clé secrète  $K$ . Pour envoyer à Bob un message  $M$  protégé en intégrité, Alice

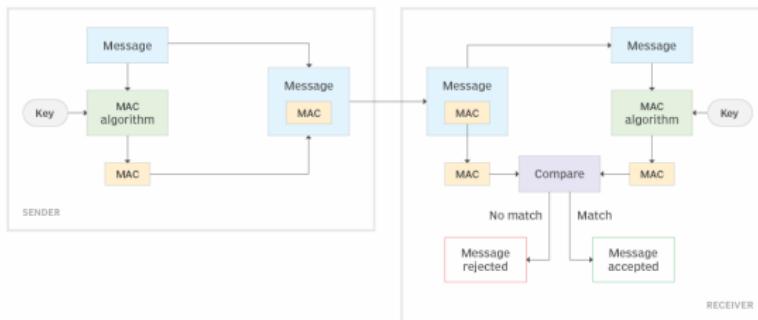
- calcule  $T = \text{MAC}_K(M)$
- transmet  $(M, T)$  à Bob

Pour vérifier l'intégrité, Bob :

- reçoit  $(M^*, T^*)$  potentiellement différent de  $(M, T)$
- calcule  $\text{MAC}_K(M^*)$ , et teste  $\text{MAC}_K(M^*)? = T^*$
- si oui, alors (avec une écrasante probabilité)  $(M^*, T^*) = (M, T)$  et Bob déclare l'intégrité valide
- si non, alors (sûrement)  $(M^*, T^*) \neq (M, T)$  et Bob déclare l'intégrité invalide

# Un p'tit dessin

## Generating and verifying message authentication codes



# Intégrité symétrique. Hachage

1 Intégrité symétrique. MAC

2 Sécurité d'un MAC

3 Fonction de hachage

4 Constructions de MAC

5 Intégrité et confidentialité combinées

# Sécurité d'un MAC

Principe : l'adversaire connaît des couples

$(M_1, T_1 = \text{MAC}_K(M_1)), \dots, (M_q, T_q = \text{MAC}_K(M_q))$ , doit **contrefaire** un nouveau MAC, i.e. trouver une paire  $(M, T = \text{MAC}_K(m))$  pour  $m \neq M_1, \dots, M_q$ .

Contexte : les  $M_i$  sont

- connus sans contrôle : attaque à *messages connus* (*Known Message Attack*, KMA)
- choisis par l'adversaire. Les  $T_i$  sont obtenus par requêtes à un oracle de MAC : attaque à *messages choisis* (*Chosen Message Attack*, CMA)

Défi :

- *contrefaçon universelle* : l'adversaire trouve  $T$  pour tout  $M$   
→ recouvrement de clé
- *contrefaçon selective* : l'adversaire trouve  $T$  pour un message spécifique  $M$ , qu'il a éventuellement choisi lui-même
- *contrefaçon existentielle* : l'adversaire trouve  $T$  pour un certain message  $M$  qui n'était pas *a priori* prédictible ni ciblé avant l'attaque  
→ recouvrement de MAC

Un algorithme de MAC est considéré comme sûr s'il n'existe aucune attaque résolvant le défi avec une complexité significativement plus basse que celle des attaques génériques (brute force) :

- $2^{|K|}$  pour la contrefaçon universelle
- $2^{|T|}$  pour la contrefaçon sélective et existentielle

# Intégrité symétrique. Hachage

## 1 Intégrité symétrique. MAC

## 2 Sécurité d'un MAC

## 3 Fonction de hachage

- Définition et sécurité
- Construction Merkle-Damgård
- Construction éponge

## 4 Constructions de MAC

## 5 Intégrité et confidentialité combinées

Intégrité symétrique. MAC

Sécurité d'un MAC

**Fonction de hachage**

Constructions de MAC

Intégrité et confidentialité combinées

Définition et sécurité

Construction Merkle-Damgård

Construction éponge

# Intégrité symétrique. Hachage

## 3 Fonction de hachage

- Définition et sécurité
- Construction Merkle-Damgård
- Construction éponge

## Fonction de hachage

Fonction  $\mathcal{H}$  : prend en entrée une variable de taille quelconque, retourne une *empreinte* ou *haché* de taille fixe  $|\mathcal{H}|$ , avec les propriétés suivantes.

- Pour tout  $x$ , il est rapide de calculer  $\mathcal{H}(x)$ .
- Propriété de *sens unique*, ou *pré image résistance*  
étant donné  $y$ , il est complexe de trouver  $x$  tel que  $y = \mathcal{H}(x)$ , à moins de le connaître *a priori*.
- Propriété de *seconde pré image résistance*  
étant donné  $x_0$ , il est complexe de trouver  $x \neq x_0$  tel que  $\mathcal{H}(x) = \mathcal{H}(x_0)$ .
- Propriété de *collision résistance*  
Il est complexe de trouver  $x_1 \neq x_2$  tels que  $\mathcal{H}(x_1) = \mathcal{H}(x_2)$ .

## Sécurité d'une fonction de hachage

Les propriétés “sens inverse difficile” sont quantifiables. Les complexités génériques sont :

- pré image :  $O(2^{|\mathcal{H}|})$
- seconde pré image :  $O(2^{|\mathcal{H}|})$
- collision : **ATTENTION**  $O(\sqrt{2^{|\mathcal{H}|}}) = O(2^{|\mathcal{H}|/2})$   
⇐ paradoxe des anniversaires

Un attaquant contre l'une des propriétés sera efficace si sa complexité de succès est significativement inférieure.

Intégrité symétrique. MAC

Sécurité d'un MAC

**Fonction de hachage**

Constructions de MAC

Intégrité et confidentialité combinées

Définition et sécurité

**Construction Merkle-Damgård**

Construction éponge

# Intégrité symétrique. Hachage

## 3 Fonction de hachage

- Définition et sécurité
- **Construction Merkle-Damgård**
- Construction éponge

## Construction Merkle-Damgård

Itération d'une primitive  $f$  appelée *fonction de compression* :

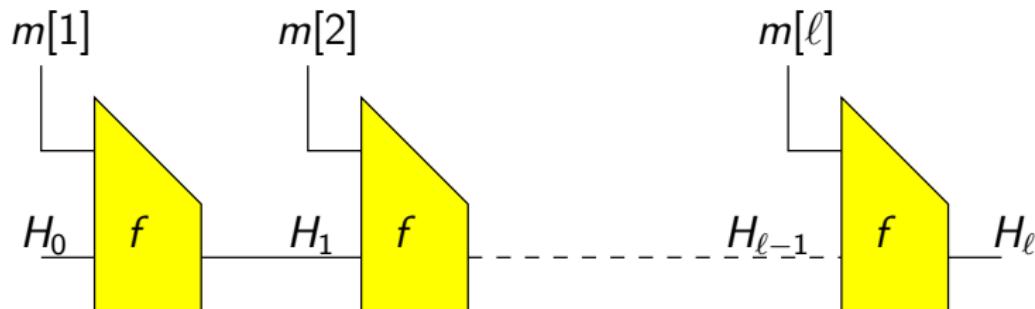
$$f : \{0,1\}^n \times \{0,1\}^h \rightarrow \{0,1\}^h$$

$n, h$  désignant des tailles prescrites à l'avance. Soit  $m$  un message.

- ① padding de  $m$  pour obtenir  $|m|$  multiple de  $n$
- ② Découper  $m$  en  $\ell = |m|/n$  blocs de  $n$  bits :  
 $m = (m[1], \dots, m[\ell])$  ;
- ③ on pose  $H_0$  une constante initiale fixe et connue ;
- ④ pour  $i = 1, \dots, \ell$ , on calcule  $H_i = f(m[i], H_{i-1})$  ;
- ⑤ retourner  $H_\ell$ .

Tout se passe comme si le message était “absorbé” bloc par bloc par une machine à état interne, puis “digéré” au sens où chaque bloc absorbé intervient dans la mise à jour l'état interne.

# Merkle-Damgård avec un dessin



# Exemples de fonctions de hachage Merkle-Damgård

<b>Algorithmes</b>		<b>Taille état interne</b>	<b>Taille haché</b>	<b>Taille bloc</b>
MD4, MD5		128	128	512
RIPEMD-128		128	128	512
RIPEMD-160		160	160	
RIPEMD-256		128	256	
RIPEMD-320		160	320	
SHA-0, SHA-1		160	160	512
SHA-2	SHA-224	256	224	512
	SHA-256		256	
	SHA-384	512	384	1 024
	SHA-512		512	
	SHA-512/224		224	
	SHA-512/256		256	

# Intégrité symétrique. Hachage

## 3 Fonction de hachage

- Définition et sécurité
- Construction Merkle-Damgård
- **Construction éponge**

## Construction éponge

Itération d'une primitive bijective à sens unique  $f$  sur  $r + c$  bits.

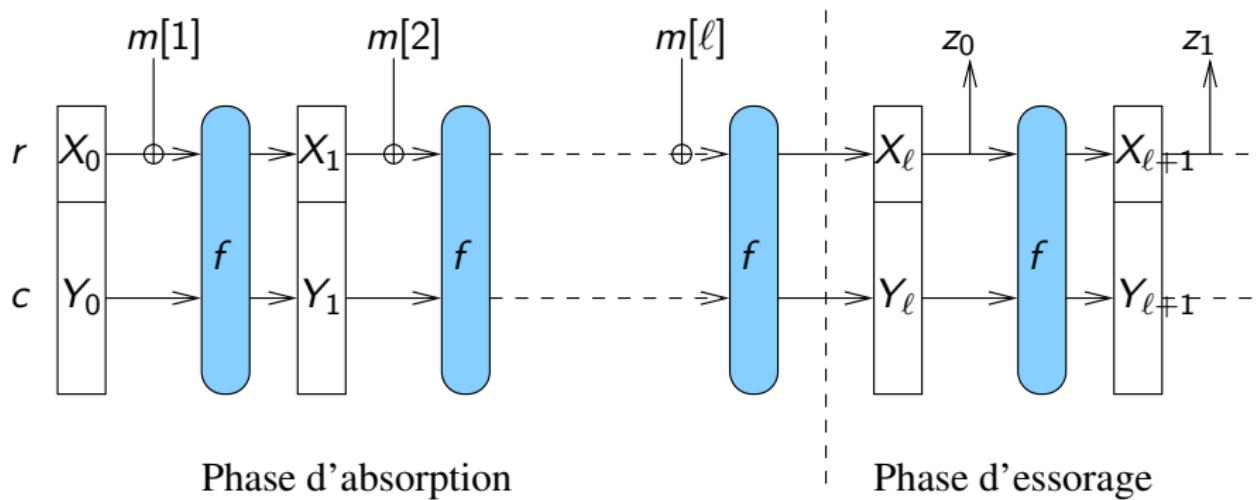
- $r$  est le *taux* de la fonction
- $c$  est la *capacité* de la fonction
- la taille du haché  $h$  est un (petit) multiple de  $r$ .

On note  $(X, Y)$  l'état interne de la fonction, avec  $|X| = r$  et  $|Y| = c$

## Séquencement

- ① padding de  $m$  pour obtenir  $|m|$  multiple de  $r$
- ② Découper  $m$  en  $\ell = |m|/r$  blocs de  $r$  bits  
 $m = (m[1], \dots, m[\ell])$  ;
- ③ on pose  $(X_0, Y_0) = (0, 0)$
- ④ phase d'absorption :  
pour  $i = 1, \dots, \ell$ ,  $(X_i, Y_i) = f(X_{i-1} \oplus m[i], Y_{i-1})$
- ⑤ phase d'essorage :  
pour  $j = \ell + 1, \dots, \ell + h/r$ ,  $(X_j, Y_j) = f(X_j, Y_j)$
- ⑥ retourner  $(z_0, \dots, z_{h/r-1}) = (X_{\ell+1}, \dots, X_{\ell+h/r})$

## Construction éponge avec un dessin



Phase d'absorption

Phase d'essorage

# Exemples de fonctions de hachage éponge

C'est le standard actuel de hachage

<b>Algorithmes</b>		<b>Taille état interne</b>	<b>Taille haché</b>	<b>Taille bloc</b>
SHA-3 (Keccak)	SHA3-224	256	224	512
	SHA3-256		256	
	SHA3-384	512	384	1 024
	SHA3-512		512	
	SHA3-512/224		224	
	SHA3-512/256		256	

# Intégrité symétrique. Hachage

## 1 Intégrité symétrique. MAC

## 2 Sécurité d'un MAC

## 3 Fonction de hachage

## 4 Constructions de MAC

- Avec une fonction de hachage
- Avec une primitive bloc

## 5 Intégrité et confidentialité combinées

# Intégrité symétrique. Hachage

## 4 Constructions de MAC

- Avec une fonction de hachage
- Avec une primitive bloc

## MAC avec une fonction de hachage

On se donne une fonction de hachage, de taille de bloc  $b$ , de taille de haché  $h$ , et une clé secrète  $K$ .

On suppose  $h, |K| < b$ .

Toutes les méthodes incluent un prétraitement qui consiste à :

- padder la clé  $K$  en  $K_{\text{pad}}$  de sorte que  $|K_{\text{pad}}| = b$
- padder le message  $M$  en  $M_{\text{pad}}$  de sorte que  $|M_{\text{pad}}|$  soit multiple de  $b$ ,
- découper  $M_{\text{pad}}$  en  $\ell = |M_{\text{pad}}|/b$  blocs de  $b$  bits  $(M_{\text{pad}}[0], \dots, M_{\text{pad}}[\ell - 1])$

## Les meilleures méthodes

### Méthode HMAC

- padder  $K$  en  $K_{\text{pad}}$  et  $M$  en  $M_{\text{pad}}$
- calculer  $y = \mathcal{H}(K_{\text{pad}} \oplus \text{ipad} \parallel M_{\text{pad}}[0] \parallel \dots \parallel M_{\text{pad}}[\ell - 1])$
- padder  $y$  en  $y_{\text{pad}}$  de sorte que  $|y_{\text{pad}}| = b$
- calculer  $T = \mathcal{H}(K_{\text{pad}} \oplus \text{opad} \parallel y)$ .
- retourner  $T$

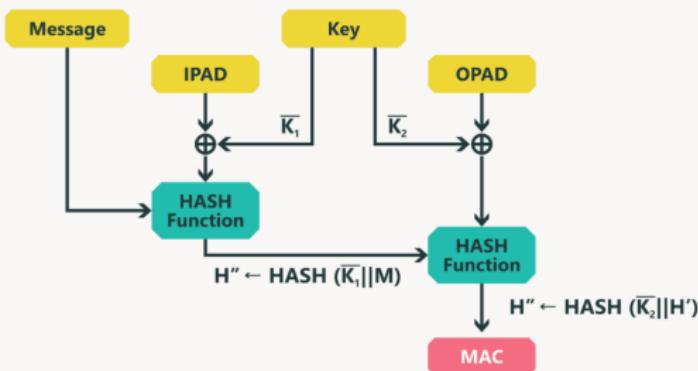
$\text{ipad} = \underbrace{0x36\dots36}_{b \text{ bits}}, \text{opad} = \underbrace{0x5c\dots5c}_{b \text{ bits}}$

### Méthode de l'enveloppe

- padder  $K$  en  $K_{\text{pad}}$  et  $M$  en  $M_{\text{pad}}$
- calculer  $T = \mathcal{H}(K_{\text{pad}} \parallel M_{\text{pad}}[0] \parallel \dots \parallel M_{\text{pad}}[\ell - 1] \parallel K_{\text{pad}})$
- retourner  $T$

# HMAC en image

## HASH-BASED MESSAGE AUTHENTICATION CODE (HMAC)



## Les "oui mais"

### Méthode du préfixe :

- padder  $K$  en  $K_{\text{pad}}$  et  $M$  en  $M_{\text{pad}}$
- calculer  $T = \mathcal{H}(K_{\text{pad}} \parallel M_{\text{pad}}[0] \parallel \dots \parallel M_{\text{pad}}[\ell - 1])$
- retourner  $T$

Sûr seulement pour messages de taille fixe. Sinon, contrefaçon (presque) universelle possible avec complexité négligeable sous CMA avec une requête.

### Méthode du suffixe

- padder  $K$  en  $K_{\text{pad}}$  et  $M$  en  $M_{\text{pad}}$
- calculer  $T = \mathcal{H}(M_{\text{pad}}[0] \parallel \dots \parallel M_{\text{pad}}[\ell - 1] \parallel K_{\text{pad}})$
- retourner  $T$

Contrefaçon existentielle possible avec complexité  $O(2^{h/2})$  sous CMA avec une requête.

# Intégrité symétrique. Hachage

## 4 Constructions de MAC

- Avec une fonction de hachage
- Avec une primitive bloc

# MAC avec une primitive bloc

On se donne une primitive bloc  $E_K$  de taille de bloc  $b$ .

Deux approches :

- les modes par chainage : CBCMAC, EMAC, OMAC, CMAC, etc.
- les modes par combinaison (plus rare) : PMAC, etc.

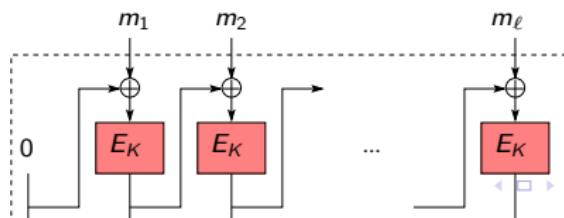
# CBCMAC

Sûr seulement lorsque messages de taille fixe.

- padder  $M$  et découper le message paddé en  $M[1] \parallel \dots \parallel M[\ell]$
- calcul chaîné “façon CBC”

$$\begin{aligned} t[1] &= E_K(M[1] \oplus \underbrace{0 \dots 0}_{b \text{ bits}}) \\ t[2] &= E_K(M[2] \oplus t[1]) \\ &\dots \\ t[\ell] &= E_K(M[\ell] \oplus t[\ell - 1]) \end{aligned}$$

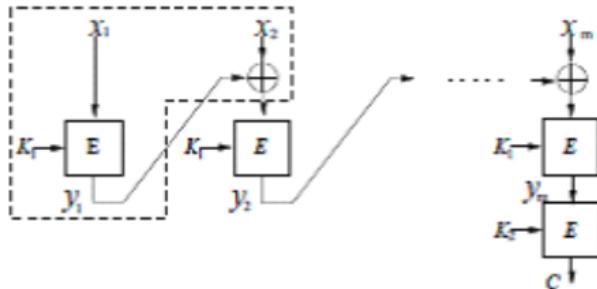
- retourner  $T := t[\ell]$ , et détruire les valeurs intermédiaires  $t[1], \dots, t[\ell - 1]$ .



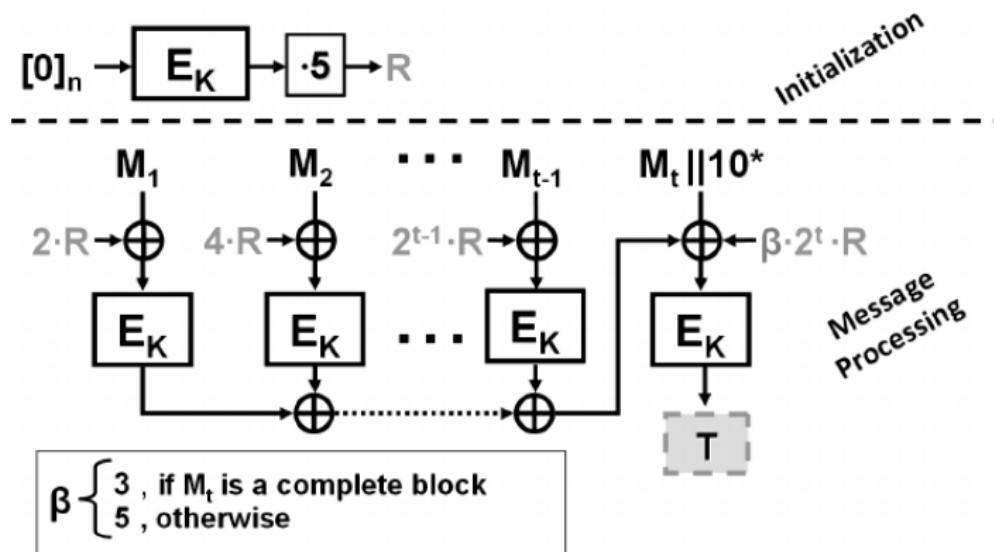
## EMAC (Encrypted MAC)

Une variante pour messages de taille variable. Utilise deux clés secrètes  $K, K'$ .

- calculer  $t = \text{CBCMAC}_K(M)$
- calculer  $\text{EMAC}_{K,K'}(M) := T = E_{K'}(t)$



## PMAC (Parallelizable MAC)



# Intégrité symétrique. Hachage

1 Intégrité symétrique. MAC

2 Sécurité d'un MAC

3 Fonction de hachage

4 Constructions de MAC

5 Intégrité et confidentialité combinées

- Version de base
- Version AEAD

# Intégrité symétrique. Hachage

## 5 Intégrité et confidentialité combinées

- Version de base
- Version AEAD

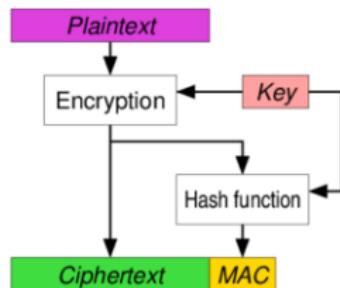
# Intégrité et confidentialité

Essentiellement deux approches :

- Par composition d'un chiffrement symétrique et d'un MAC (voire hachage sans clé) : CBC 2 passes, CCM, GCM
  - ces algorithmes sont dits "en deux passes"
  - ils requièrent deux clés  $K_{\text{conf}}$  et  $K_{\text{int}}$  qu'il est fortement conseillé de prendre différentes.
- L'approche "en une passe", ou "deux-en-un" : IAPM, OCB
  - on part d'un chiffrement auquel on adjoint une procédure légère
  - on récupère l'intégrité pour un faible surcout calculatoire
  - une seule clé est nécessaire
  - **MAIS ATTENTION** : la "procédure légère" qui assure l'intégrité en bonus **ne peut pas être extraite et isolée** pour en faire un MAC.

# Encrypt-Then-MAC

- calculer  $C = \text{SymEnc}_{K_{\text{conf}}}(M)$
- calculer  $T = \text{MAC}_{K_{\text{int}}}(C)$
- retourner  $C, T$

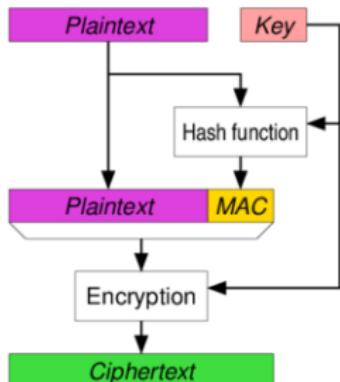


Bonus : renforcement de la confidentialité

Le mode combiné est IND-CCA si SymEnc est IND-CPA

## MAC-then-Encrypt

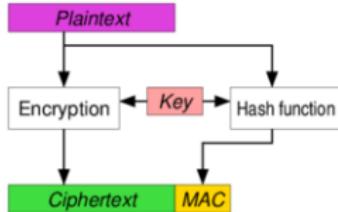
- calculer  $T = \text{MAC}_{K_{\text{int}}}(M)$
- calculer  $C = \text{SymEnc}_{K_{\text{conf}}}(M, T)$
- retourner  $C$



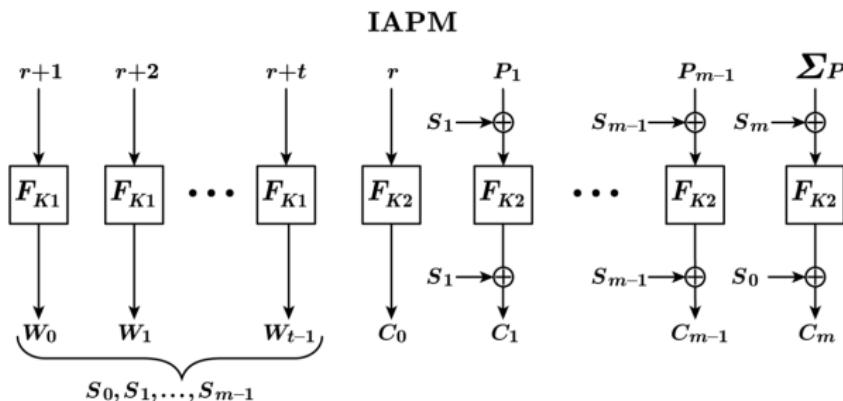
# Encrypt-And-MAC

Dangereux car pourrait ne pas assurer la confidentialité. (En pratique, cela n'arrive pas, mais le cas est théoriquement possible)

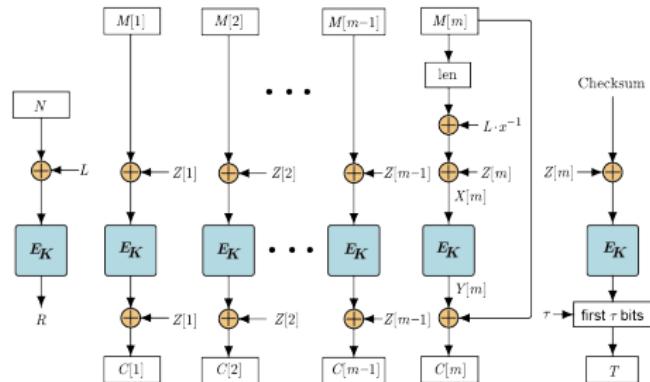
- calculer  $C = \text{SymEnc}_{K_{\text{conf}}}(M)$
- calculer  $T = \text{MAC}_{K_{\text{int}}}(M)$
- retourner  $C, T$



# IAPM (Integrity Aware Parallelizable Mode)



# OCB (Offset codebook mode)



# Intégrité symétrique. Hachage

## 5 Intégrité et confidentialité combinées

- Version de base
- Version AEAD

## Les algorithmes AEAD

*Authenticated Encryption with Additional Data*, chiffrement intégré avec données additionnelles

- la confidentialité porte sur un texte clair
- l'intégrité porte sur le même texte clair ET sur d'autres données transmises non chiffrées, comme des données d'entête non sensibles

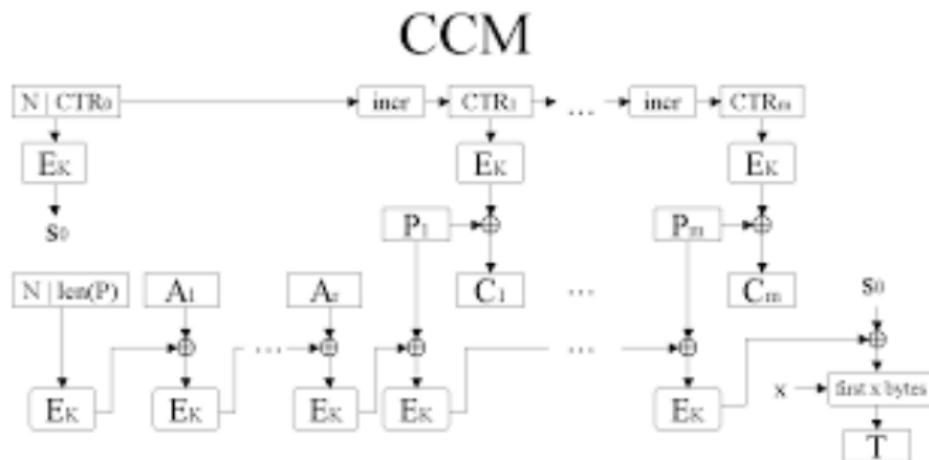
Fonctionnement typique :

---

Entrées :  $K_{\text{conf}}$  et  $K_{\text{int}}$ , entête  $H$ , clair  $P$

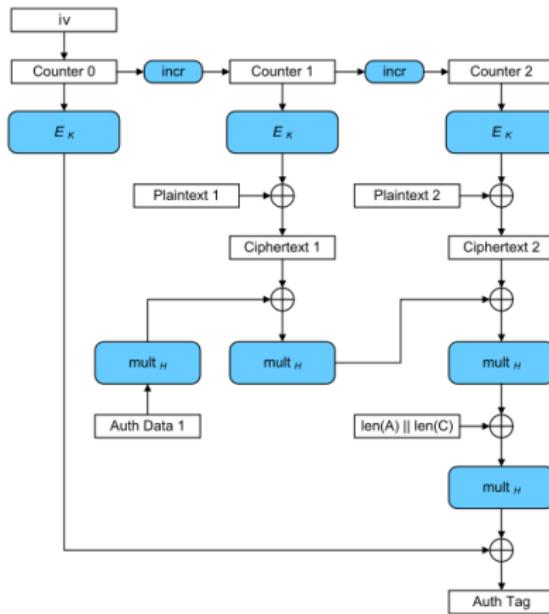
- 1 : chiffré  $C = E_{K_{\text{conf}}}(P)$
  - 2 : tag  $T = \text{MAC}_{K_{\text{int}}}(H||C)$
  - 3 : retourner  $(H, C, T)$
-

## CCM (Counter Mode with CBCMAC)



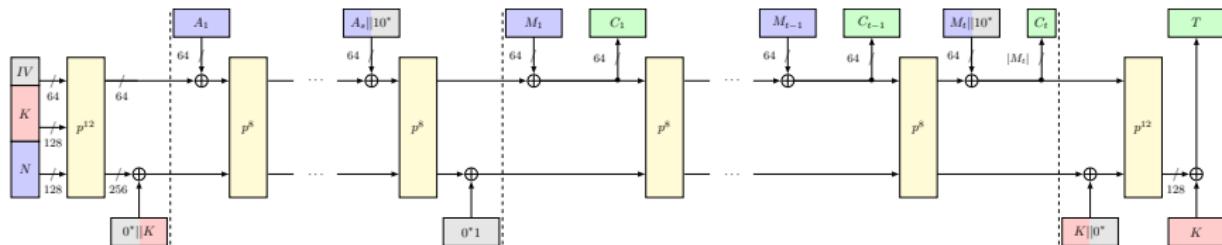
# GCM (en version AEAD)

## Galois Counter Mode

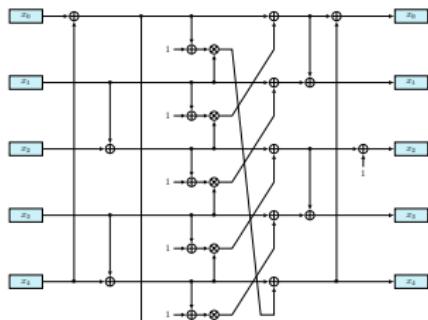


# ASCON

## Schéma général



Sbox :



## – Partie 6 – Signature

# Signature

1 Principe

2 Sécurité des signatures

3 Exemples

# Principe

Analogue attendu de la signature papier. Elle offre trois services cryptographiques :

- authentification du signataire
- intégrité des données
- vérification possible devant une tierce partie de confiance, et donc *non-répudiation* du signataire.

Une seule personne peut signer. Tout le monde peut vérifier.

Utiliser la cryptographie asymétrique.

## Mise en œuvre

Il y a deux fonctions “inverses” l’une à l’autre :

- une fonction de signature  $\text{Sig}$  qui à un message de taille quelconque, associe un motif  $\sigma$  de taille fixe appelé signature
- une fonction de vérification  $\text{Ver}$  qui, à un couple (message, signature) retourne un booléen “vrai” ou “faux”

Chaque utilisateur possède une clé se décomposant en deux parties :

- une partie devant rester secrète notée  $K_{\text{sig}}$  et appelée *clé (privée) de signature*.
- une partie publique notée  $K_{\text{ver}}$  et appelée *clé (publique) de vérification* qui doit être intègre.

Propriété fondamentale :

$$\text{Ver}_{K_{\text{ver}}}(M, \sigma) = \text{vrai} \Leftrightarrow \sigma = \text{Sig}_{K_{\text{sig}}}(M)$$

Ainsi  $K_{\text{sig}}$  et  $K_{\text{ver}}$  sont mathématiquement reliées. Mais il doit être calculatoirement impossible de déduire  $K_{\text{sig}}$  de  $K_{\text{ver}}$  et d'autres paramètres publics.

## Schéma général de la transmission d'un message signé

Pour envoyer un message  $M$  signé à Bob, Alice :

- calcule  $\sigma = \text{Sig}_{K_{\text{sig}}}(M)$
- Alice transmet  $(M, \sigma)$  à Bob

Pour vérifier la signature, Bob :

- reçoit  $(M^*, \sigma^*)$ , potentiellement différent de  $(M, \sigma)$ .
- calcule la valeur booléenne  $b = \text{Ver}_{K_{\text{ver}}}(M^*, \sigma^*)$
- si  $b = \text{vrai}$ , alors (avec écrasante probabilité) Bob déclare la signature valide, et est convaincu des trois propriétés de sécurité
- si  $b = \text{faux}$ , alors (sûrement) Bob déclare la signature invalide.

**ATTENTION** : contrairement au cas symétrique (intégrité par un MAC),

- la vérification **ne doit pas consister** à recalculer la signature d'Alice sur le message  $M^*$  et la comparer avec  $\sigma^*$ .
- Cela **irait à l'encontre** du principe même de la signature.

# Primitives

Mise en œuvre pratique d'une signature :

- ① hachage du message
  - ② traitement du haché par une primitive de type "problème mathématique difficile" (RSA, Elgamal, ...)
- on peut signer des messages de taille quelconque
  - les primitives de type "problème mathématique difficile" traitent des données de taille fixe

Le hachage permet de "calibrer" la valeur à laquelle on applique l'asymétrique.

# Signature

1 Principe

2 Sécurité des signatures

3 Exemples

## Sécurité d'une signature

Le modèle est calqué sur celui de la sécurité des MAC.

Principe : l'adversaire connaît des couples

$(M_1, \sigma_1 = \text{Sig}_{K_{\text{sig}}}(M_1)), \dots, (M_q, \sigma_q = \text{Sig}_{K_{\text{sig}}}(M_q))$ , doit *contrefaire* une nouvelle signature, i.e. trouver une paire  $(M, \sigma = \text{Sig}_{K_{\text{sig}}}(M))$  pour  $M \neq M_1, \dots, M_q$ .

Contexte : les  $M_i$  sont

- connus sans contrôle : attaque à *messages connus* (*Known Message Attack*, KMA)
- choisis par l'adversaire. Les  $\sigma_i$  sont obtenus par requêtes à un oracle de signature : attaque à *messages choisis* (*Chosen Message Attack*, CMA)

Défi :

- *contrefaçon universelle* : l'adversaire trouve  $\sigma$  pour tout  $M$   
( $\Leftrightarrow$  recouvrement de clé)
- *contrefaçon sélective* : l'adversaire trouve  $\sigma$  pour un message spécifique  $M$ , qu'il a éventuellement choisi lui-même
- *contrefaçon existentielle* : l'adversaire trouve  $\sigma$  pour un certain message  $M$  qui n'était pas *a priori* prédictible ni ciblé avant l'attaque

Un algorithme de signature est considéré comme sûr s'il n'existe aucune attaque résolvant le défi avec une complexité significativement plus basse que celle des attaques génériques :

- résolution du problème mathématique sous-jacent pour recouvrir la clé privée : complexité grande (rappel : néanmoins  $\ll 2^{|K_{\text{sig}}|}$ )  
⇒ contrefaçon universelle
- recouvrement de signature : complexité  $2^{|\sigma|}$
- trouver une seconde préimage à la fonction de hachage  $2^{|\mathcal{H}|}$   
⇒ contrefaçon sélective
- trouver une collision à la fonction de hachage  $\sqrt{2^{|\mathcal{H}|}}$   
⇒ contrefaçon existentielle (voire sélective)

# Signature

## 1 Principe

## 2 Sécurité des signatures

## 3 Exemples

- Signature RSA et factorisation
- Signature et logarithme discret

# Signature

## 3 Exemples

- Signature RSA et factorisation
- Signature et logarithme discret

# Signature RSA

$\mathcal{H}$  est une fonction de hachage publique et commune.

## Génération des clés

- génération de deux nombres premiers  $p$  et  $q$  ;
- calcul de  $n = pq$
- choix d'un entier  $e$  premier avec  $\phi(n) = (p - 1)(q - 1)$
- calcul de  $d$  inverse de  $e$  modulo  $\phi(n)$ .  
Ainsi  $ed \equiv 1 \pmod{\phi(n)}$
- Clé publique :  $(n, e)$ . Clé privée :  $(p, q, d)$

## Signature

- $\sigma = \text{RSA-Sig}_{K_{\text{sig}}}(M) = \mathcal{H}(M)^d \bmod n$

## Vérification

- $\text{RSA-Ver}_{K_{\text{ver}}}(M, \sigma) = \text{true} \Leftrightarrow \mathcal{H}(m) = \sigma^e \bmod n$

## Sécurité

- Basée sur la difficulté de la factorisation de  $n$ .

# Signature

## 3 Exemples

- Signature RSA et factorisation
- Signature et logarithme discret

# Signature Elgamal et logarithme discret sur $\mathbb{F}_p^*$

$\mathcal{H}$  est une fonction de hachage publique et commune.

## Génération des clés

- génération un nombre premier  $p$ ,
- choix d'un générateur  $g$  du groupe multiplicatif  $\mathbb{F}_p^*$ ,
- choix d'un entier  $K_{pr} \in \{1, \dots, p - 2\}$
- calcul de  $h = g^{K_{pr}} \text{ mod } p$
- Clé publique :  $(p, g, h)$ . Clé privée :  $K_{pr}$ .

Signature de  $M$  :

- choisir un entier aléatoire secret  $r \in \{1, \dots, p - 2\}$  inversible modulo  $p - 1$ .
- calculer  $\sigma_1 = g^r \bmod p$
- calculer  $\sigma_2 = (\mathcal{H}(M) - K_{pr}\sigma_1)r^{-1} \bmod (p - 1)$
- si  $\sigma_2 = 0$  recommencer
- retourner  $(\sigma_1, \sigma_2)$

Vérification de  $(M, (\sigma_1, \sigma_2))$  :

- vérifier  $1 \leq \sigma_1 \leq p - 1$  et  $1 \leq \sigma_2 \leq p - 2$
- vérifier  $g^{\mathcal{H}(M)} = h^{\sigma_1} \sigma_1^{\sigma_2} \bmod p$

# DSA

Variante de la signature Elgamal, travaillant dans un grand sous-groupe (cyclique) de  $\mathbb{F}_p^*$ .

## Génération des clés

- générer un nombre premier  $q$ , puis un autre premier  $p$  tel que  $q$  divise  $p - 1$
- choisir un élément  $g \in \mathbb{F}_p^*$  d'ordre  $q$  (on travaille dans le sous-groupe de  $\mathbb{F}_p^*$  engendré par  $g$ )
- choisir un entier  $K_{pr} \in \{1, \dots, q - 1\}$
- calculer  $h = g^{K_{pr}} \bmod p$
- Clé publique :  $(p, g, h)$ . Clé privée :  $K_{pr}$ .

Signature de  $M$  :

- choisir un entier aléatoire secret  $r \in \{1, \dots, q - 1\}$  (donc inversible modulo  $q$ ).
- calculer  $\sigma_1 = (g^r \bmod p) \bmod q$
- calculer  $\sigma_2 = (\mathcal{H}(M) + K_{pr}\sigma_1)r^{-1} \bmod q$
- si  $\sigma_1 = 0$  ou  $\sigma_2 = 0$  recommencer
- retourner  $(\sigma_1, \sigma_2)$

Vérification de  $(M, (\sigma_1, \sigma_2))$  :

- vérifier  $1 \leq \sigma_1, \sigma_2 \leq q - 1$ .
- vérifier  $\sigma_1^{\sigma_2} = g^{\mathcal{H}(M)} h^{\sigma_1} \bmod p$

# ECDSA

Analogue du DSA : on travaille dans un grand sous-groupe cyclique d'une courbe elliptique

## Génération des clés

- une courbe elliptique  $\mathbb{E}(\mathbb{F}_p)$ ,  $p$  premier
- un point  $P \in \mathbb{E}$  d'ordre  $n$
- un entier  $K_{pr} \in \{1, \dots, n - 1\}$
- le point  $Q = K_{pr}P \in \mathbb{E}$
- Clé publique :  $(p, \mathbb{E}, P, n, Q)$ . Clé privée :  $K_{pr}$

Signature de  $M$  :

- choisir un entier aléatoire secret  $r \in \{1, \dots, n - 1\}$
- calculer  $\sigma_1 = x_r P \pmod{n}$
- calculer  $\sigma_2 = r^{-1} (\mathcal{H}(M) + \sigma_1 K_{pr}) \pmod{n}$
- si  $\sigma_1 = 0$  ou  $\sigma_2 = 0$ , recommencer
- retourner  $(\sigma_1, \sigma_2)$  ou  $(\sigma_1, -\sigma_2)$

Vérification de  $(M, (\sigma_1, \sigma_2))$  :

- vérifier  $1 \leq \sigma_1, \sigma_2 \leq n - 1$
- calculer le point  $R = (\mathcal{H}(M)\sigma_2^{-1})P + (\sigma_1\sigma_2^{-1})Q$ .
- vérifier que  $R \neq O_E$
- vérifier que  $\sigma_1 = x_R$

## – Partie 7 – Authentification

# Protocoles d'authentification

Problème :

Alice veut s'authentifier auprès de Bob

Propriétés de base :

- Le protocole doit être réussir si Alice et Bob sont honnêtes
- Si Alice est malhonnête le protocole doit échouer
- Si Bob est malhonnête ne doit pas pouvoir utiliser une authentification d'Alice pour simuler une authentification d'Alice auprès d'une tierce personne Anatole

## Principe et types de solutions

Principe :

- Alice possède un secret notoirement attaché à son identité ;
- Alice prouve à Bob qu'elle possède le secret sans le révéler.

Types :

- authentification faible : le secret est un mot de passe ;
- authentification forte : par challenge-réponse ;
- authentification *zero knowledge* : aucune usure théorique du secret.

# Authentification

1 Mot de passe

2 Challenge/réponse

3 Zero Knowledge

## Authentification par mot de passe

Alice possède un mot de passe secret. Bob est **supposé honnête** (système, serveur de confiance)

À ÉVITER : mots de passe stockés en clair dans le système !

Utilisation d'une fonction de hachage

- Bob stocke les hachés des mots de passe
- Authentification
  - Alice entre son mot de passe.
  - Bob hache l'entrée d'Alice et vérifie la conformité

Attaques :

- par dictionnaire (hors ligne) si le nombre de mots de passe est trop petit
- par rejeu si saisie du mot de passe mal protégée (attaque physique)

## One Time Password

Alice :

- choisit une constante  $t$  définissant le nombre d'authentifications
- choisit une fonction à sens unique  $H$  publique
- choisit un secret  $w$
- transfère  $w_0 = H^t(w)$  à Bob

Bob :

- initialise un compteur :  $\text{cpt} \leftarrow 1$
- Authentification numéro  $i$ 
  - Alice calcule  $w_i = H^{t-i}(w)$  et transmet  $(i, w_i)$  à Bob
  - Bob vérifie  $i = \text{cpt}$  et  $w_{i-1} = H(w_i)$ , puis met à jour le compteur  $\text{cpt} \leftarrow \text{cpt} + 1$

# Authentification

1 Mot de passe

2 Challenge/réponse

3 Zero Knowledge

# Authentification forte : principe du challenge

## Principe

- Alice prétend posséder un secret
- Bob envoie un challenge à Alice
- Alice résout le challenge et convainc Bob qu'elle est bien détentrice du secret prétendu

Avec un chiffrement asymétrique :

- Bob choisit un aléa  $X$  et envoie  $Y = E_K(X)$  à Alice.
- Alice renvoie à Bob  $Z = E_K^{-1}(Y)$
- Bob vérifie  $X = Z$  et si oui est convaincu qu'Alice possède la clé de déchiffrement

Variante avec signature asymétrique :

- Bob choisit un aléa  $X$  et l'envoie à Alice.
- Alice renvoie à Bob  $Y = S_K(X)$
- Bob vérifie  $V_K(X, Y)$  et si oui est convaincu qu'Alice possède la clé de signature

# Authentification

- 1 Mot de passe
- 2 Challenge/réponse
- 3 Zero Knowledge

## Protocoles *Zero Knowledge*

Protocoles à “apport nul de connaissance”

- Alice va prouver qu'elle possède un secret sans révéler la moindre information sur ce secret

Schéma général :

- Alice choisit un aléa
- et le met en gage en transmettant un témoin à Bob
- Bob choisit un challenge aléatoire et l'envoie à Alice
- Alice répond au challenge et envoie la réponse à Bob

## Pourquoi Zero Knowledge ?

- Le protocole se déroule en plusieurs passes ;
- Alice n'a pas besoin d'utiliser son secret à toutes les passes pour répondre au challenge ;
- Bob est convaincu à la fin du protocole ;
- MAIS Bob ne peut pas transférer cette conviction et aucun autre spectateur du protocole ne peut être convaincu.

## Allégorie de la caverne

- Alice choisit au hasard d'entrer par la gauche ou la droite
- Bob entre quand Alice est hors de vue
- Bob choisit au hasard un couloir et demande à Alice de sortir par ce couloir

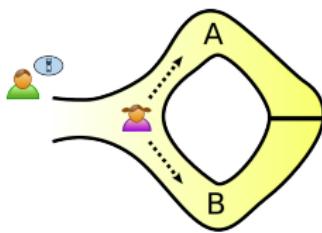


Figure – Alice entre

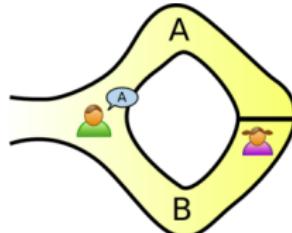


Figure – Bob l'appelle

## La notion de simulateur et Zero Knowledge

Si Anatole (usurpateur d'Alice) et Bob sont complices. Bob place une caméra à l'entrée de la caverne, et fabrique un film factice où seules figurent les passes "succès" (il y en a la moitié en moyenne).

Un spectateur ne peut pas faire la différence entre un vrai film et un film factice. Ce dernier a été fabriqué sans le secret d'Alice.

## Exemple : protocole de Fiat-Shamir

Alice possède  $n = pq$  un entier RSA,  $s$  secret et  $v$  public tels que  $s^2 = v \bmod n$ ,

### Protocole

- Alice génère un aléa  $r$  et (mise en gage) transmet  $x = r^2 \bmod n$  à Bob
- Bob choisit aléatoirement un bit  $b$  challenge et le transmet à Alice
- Alice calcule  $y = r/s^b \bmod n$  et le transmet à Bob
- Bob vérifie  $y^2 v^b = x \bmod n$

## Sécurité :

- Si Alice ne connaissait pas  $s$ , elle aurait une chance sur 2 de bien répondre.
- En itérant  $t$  fois l'échange, la probabilité de succès d'Alice malhonnête est de  $2^{-t}$ .

## Zero Knowledge :

- si Alice et Bob sont malhonnêtes et complices, il leur est possible de produire le “film” d'une authentification réussie : ils se filment et effacent les scènes qui ont échoué (en moyenne une sur deux) ;
- ce “film malhonnête” est indiscernable du film d'une authentification réussie honnêtement.
- le film malhonnête n'apporte aucune information sur le secret puisqu'il a été construit sans le secret.

– Partie 8 –  
Quantique et cryptographie

# Quantique et cryptographie

1 Ne pas confondre

2 Cryptographie quantique

3 Cryptographie post quantique

# Ne pas confondre

- *Crypto quantique*

- Consiste à concevoir des algorithmes cryptographiques mettant en jeu des objets quantiques
- la sécurité repose sur les lois de la mécanique quantique, et non plus (seulement) sur des résultats mathématiques
- C'est une **alternative** à la cryptographie classique

- *Cryptographie post-quantique*

- Consiste à concevoir des algorithmes cryptographiques - classiques ou quantiques -
- l'évaluation tient compte de la puissance d'un potentiel *ordinateur quantique*.

# Quantique et cryptographie

1 Ne pas confondre

2 Cryptographie quantique

3 Cryptographie post quantique

## Exemple de l'échange de clé quantique

L'exemple emblématique de l'échange de clé est celui de Bennett-Brassard (1984)

Au départ, Alice et Bob possèdent

- des objets qui suivent les lois de la mécanique quantique
- un canal de communication quantique
- un canal de communication classique authentifié

## Principe

- phase de transmission,
  - Alice envoie une suite de q-bits à Bob, codés selon une base de polarisation choisie au hasard parmi deux.
  - A la fin de cette phase, Alice et Bob ont chacun un jeu de données corrélées l'une à l'autre, mais pas encore de secret commun
- phase d'extraction de la clé
  - Alice envoie à Bob sur le canal authentifié une information additionnelle.
  - de laquelle Alice et Bob déduisent lesquels des q-bits ont été effectivement transmis de manière sécurisée

# Quantique et cryptographie

1 Ne pas confondre

2 Cryptographie quantique

3 Cryptographie post quantique

- Impact sur la cryptographie symétrique
- Impact sur la cryptographie asymétrique

# Ordinateur quantique et cryptographie

- Un ordinateur quantique, s'il existait, pourrait réaliser certaines opérations beaucoup plus vite qu'un ordinateur classique.
- Certaines attaques cryptographiques seraient beaucoup plus efficaces.

Le défi technologique est d'arriver à créer un tel ordinateur quantique avec un état quantique suffisamment grand pour accéder aux dimensionnements des algorithmes cryptographiques standard

# Quantique et cryptographie

## 3 Cryptographie post quantique

- Impact sur la cryptographie symétrique
- Impact sur la cryptographie asymétrique

## L'algorithme de Grover

Permet une recherche d'élément en  $O(\sqrt{N})$  dans un ensemble de cardinal  $N$ .

- Ensemble = liste **non triée**
- Un ordinateur classique ne fait pas mieux que  $O(N)$  par parcours exhaustif

Il permet de rendre toute recherche exhaustive de clé accessible en complexité  $O\left(\sqrt{2^{|clé|}}\right)$ .

En cryptographie symétrique, il faudrait **doubler la taille des clés** pour conserver un même niveau de sécurité. Cet impact est aujourd'hui considéré comme mineur.

# Quantique et cryptographie

## 3 Cryptographie post quantique

- Impact sur la cryptographie symétrique
- Impact sur la cryptographie asymétrique

# L'algorithme de Shor : la fin de la factorisation et du log discret

Permet de

- factoriser un nombre composé  $n$  en complexité de  $O((\log n)^3)$
- résoudre le problème du logarithme discret en temps polynomial en  $\log n$

Impact

- la complexité du cassage serait d'un ordre de grandeur comparable à celle de la mise en œuvre légitime de l'algorithme, ce qui deviendrait un non-sens.
- par exemple, pour la factorisation, pour conserver un niveau de sécurité de  $2^{128}$ , il faudrait que la taille du module  $n$  passe de 3 072 bits à ... $2^{43}$  bits (environ un téra-octet).

## Exemples d'alternatives

- Cryptographie basée sur les réseaux euclidiens, et notamment le problème (difficile) de la recherche de vecteurs courts.
- Cryptographie basée sur les codes correcteurs, et notamment le problème (difficile) du décodage dans un code aléatoire
- Cryptographie basée sur la difficulté de résoudre des systèmes d'équations polynomiales multivariées sur un corps fini

# Références

## Références (1/2)

- ① Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.
- ② Douglas Stinson, *Cryptographie : théorie et pratique*, traduction de Serge Vaudenay, International Thomson Publishing France, 1996.
- ③ Bruce Schneier, *Cryptographie appliquée*, 2ème édition, traduction de Laurent Viennot, International Thomson Publishing France, 1997.
- ④ Rudolf Lidl, Harald Niederreiter, *Introduction to finite fields and applications*, Cambridge University Press, 1986.
- ⑤ F. J. MacWilliams, N.J.A. Sloane, *The Theory of The Error Correcting Codes*, North Holland, (première édition 1977).

## Références (2/2)

- ⑥ Site du NIST (*National Institute Standard of Technology*),  
[www.nist.gov.us](http://www.nist.gov.us).
- ⑦ Spécification de l'AES  
<https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf>
- ⑧ Animation AES-128 par E Zabala,  
<https://www.youtube.com/watch?v=gP4PqVGudtg>
- ⑨ Cours de master (24 cours d'amphi par Prof. Christof Paar)  
<https://www.youtube.com/channel/UC1usFRN4LCMcfIV7UjHNuQg/videos>
- ⑩ Recommendation for CMAC,  
<https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-38b.pdf>

## Annexes mathématiques et algorithmiques

# Annexes

## 1 Mathématiques

- Généralités
- Deux résultats de probabilités
- Algèbre générale
- Polynômes
- Arithmétique des entiers
- Arithmétique modulaire

## 2 Algorithmique

# Annexes

## 1 Mathématiques

- Généralités
  - Deux résultats de probabilités
  - Algèbre générale
  - Polynômes
  - Arithmétique des entiers
  - Arithmétique modulaire

# Notations

$\text{Card}(X)$ , $ X $ , $\#X$	cardinal de l'ensemble $X$
$\oplus$	addition modulo 2
$ x $	taille en bits d'un vecteur binaire $x$
$c_1 \parallel c_2$	concaténation des deux chaînes binaires $c_1$ et $c_2$
$v[i_1, \dots, i_r]$	$v_{i_1} \oplus \dots \oplus v_{i_r}$ , ( $v$ vecteur binaire)

# Annexes

## 1 Mathématiques

- Généralités
- Deux résultats de probabilités
- Algèbre générale
- Polynômes
- Arithmétique des entiers
- Arithmétique modulaire

# Paradoxe des anniversaires

**Expérience :** une urne contient  $n$  boules numérotées de 1 à  $n$ . On tire au hasard  $q$  boules successivement et avec remise. La probabilité dite de collision, notée  $\text{Col}(n, q)$ , qu'au moins une boule soit apparue au moins deux fois vaut

$$\begin{aligned}\text{Col}(n, q) &= 1 - \prod_{i=1}^{q-1} \left(1 - \frac{i}{n}\right) \\ &\approx 1 - \exp\left(-\frac{q(q-1)}{2n}\right) \text{ si } q \ll n\end{aligned}$$

On montre que  $\text{Col}(n, q) \geq 1/2$  dès que  $q \geq 1.17\sqrt{n}$ .

**Exemple pour les anniversaires.** La probabilité que, dans un groupe de 23 personnes, au moins deux aient la même date anniversaire est supérieure à  $1/2$ .

# Rencontre, intersection aléatoire

## Rencontre

Soit  $A$  une partie de cardinal  $a$  d'un ensemble  $E$  de cardinal  $N$ . On effectue  $q$  tirages aléatoires indépendants successifs avec remise d'un élément de  $E$ .  
Alors :

- ① la probabilité que l'un des tirages au moins ait touché  $A$  est  
 $= (1 - a/N)^q$ ;
- ② la quantité  $qa/N \ll 1$ 
  - est une bonne approximation numérique de la probabilité de rencontre si elle est  $\ll 1$ ;
  - est le nombre moyen tirages ayant touché  $A$  si elle est  $\geq 1$ .

## Cardinal moyen d'intersection aléatoire

Soient  $A$  et  $B$  deux parties d'un ensemble  $E$ , tirées aléatoirement dans l'ensemble  $\mathcal{P}(E)$  sous la contrainte des cardinaux égaux à  $a$  et  $b$  fixés à l'avance. Alors le cardinal moyen de  $A \cap B$  est égal à  $ab/N$ .

# Annexes

## 1 Mathématiques

- Généralités
- Deux résultats de probabilités
- Algèbre générale**
- Polynômes
- Arithmétique des entiers
- Arithmétique modulaire

# Groupes

Soit  $(G, \cdot)$  un groupe fini de cardinal  $n$ .

- Pour tout  $x \in G$ , il existe  $k \in \mathbb{N}$  tel que  $x^k = 1$
- le  $k$  minimum s'appelle l'*ordre* de  $x$ , noté  $\text{ord}(x)$
- $x^\ell = 1 \iff \text{ord}(x) \text{ divise } \ell$
- $x^n = 1$ , autrement dit  $\text{ord}(x)$  divise  $n$

$G$  est *cyclique* si et seulement s'il existe un élément  $g$  d'ordre  $n$ . Dans ce cas,

- $g$  s'appelle un *générateur*. Il n'est pas unique en général.
- $G = \{g^0 = 1, g^1 = g, g^2, \dots, g^{n-1}\}$
- tous les générateurs sont les  $g^\ell$  avec  $\text{pgcd}(\ell, n) = 1$ . Il y en a donc  $\phi(n)$

# Annexes

## 1 Mathématiques

- Généralités
- Deux résultats de probabilités
- Algèbre générale
- Polynômes**
- Arithmétique des entiers
- Arithmétique modulaire

# Polynômes univariés

Division euclidienne, racine

- Pour tout polynôme  $A$  et  $B$ , il existe un unique couple de polynômes  $Q, R$  tel que

$$A = BQ + R, \quad \deg(R) < \deg(B)$$

- Un élément  $a$  est racine de  $P$  si et seulement si  $X - a$  divise  $P$
- Tout polynôme de degré  $n$  admet au plus  $n$  racines

PPCM, PGCD

- Tout couple de polynômes  $(A, B)$  admettent un PPCM et un PGCD
- deux polynômes de PGCD égal à 1 sont dits premiers entre eux

# Polynômes univariés

## Identité de Bezout

- Pour tout couple de polynômes  $(A, B)$ , il existe  $U, V$  tels que  $AU + BV = \text{pgcd}(A, B)$
- théorème de Bezout : deux polynômes  $A$  et  $B$  sont premiers entre eux si et seulement s'il existe  $U, V$  tels que  $AU + BV = 1$

# Polynômes univariés

## Polynômes irréductibles

- un polynôme  $P$  de degré  $n$  est irréductible s'il ne peut pas s'écrire sous la forme  $P = QR$  avec  $1 < \deg(Q), \deg(R) < n$
- Tout polynôme se décompose de manière unique en produit de polynômes irréductibles
- les polynômes constants et ceux de degré 1 sont irréductibles
- un polynôme irréductible de degré  $\geq 2$  est sans racine.
- un polynôme de degré 2 ou 3 sans racine est irréductible
- ATTENTION : un polynôme de degré  $\geq 4$  sans racine n'est pas forcément irréductible.

## Polynômes sur un corps fini $\mathbb{F}_q$

- il existe des polynômes irréductibles de tout degré
- $P(X)$  est *primitif* s'il est irréductible et que  $X$  est un générateur du groupe multiplicatif  $\mathbb{F}_q[X]/(P(X)) - \{0\}$
- il existe des polynômes primitifs de tout degré

# Fonctions booléennes

## Vision combinatoire

- On appelle fonction booléennes de  $n$  variables toute application de  $\mathbb{F}_2^n \rightarrow \mathbb{F}_2$ .
- On peut représenter une fonction booléenne en donnant sa table de valeurs : c'est un tableau de  $2^n$  lignes et deux colonnes, où chaque ligne contient un élément de  $\mathbb{F}_2^n$  et son image par  $f$  (qui est un élément de  $\mathbb{F}_2$ ).
- L'ensemble des éléments de  $\mathbb{F}_2^n$  où  $f$  prend la valeur 1 s'appelle le support de  $f$ .
- Une fonction est équilibrée si elle prend autant de fois la valeur 0 que la valeur 1, autrement dit si son support est de cardinal  $2^{n-1}$ .

# Fonctions booléennes

## Vision algébrique

- Un *polynôme binaire* est un polynôme  $P(X_1, \dots, X_n)$  à  $n$  variables, à coefficients dans  $\{0, 1\}$  et dans lequel aucun exposant de variable n'est  $> 1$ .
- **Théorème.**

Toute fonction booléenne  $f$  à  $n$  variables peut être représentée de manière unique par un polynôme binaire à  $n$  variables :

$$\begin{aligned}f(x_1, \dots, x_n) = & a_0 + a_1 x_1 + \dots + a_n x_n \\& + a_{1,2} x_1 x_2 + \dots + a_{n-1,n} x_{n-1} x_n + \dots + a_{1,2,\dots,n} x_1 \dots x_n\end{aligned}$$

Cette écriture est la *forme algébrique normale* (ANF, *algebraic normal form*) de la fonction  $f$ .

# Annexes

## 1 Mathématiques

- Généralités
- Deux résultats de probabilités
- Algèbre générale
- Polynômes
- Arithmétique des entiers**
- Arithmétique modulaire

# Vecteurs binaires/Entiers/Hexadécimaux

Correspondance vecteurs de bits - entiers

$$\begin{array}{ccc} \{0, 1\}^b & \sim & \{0, \dots, 2^b - 1\} \\ (x_{b-1}, \dots, x_1, x_0) & \leftrightarrow & x_{b-1}2^{b-1} + \dots + x_12 + x_0 \\ & & \text{noté } [x_{b-1} \dots x_1 x_0]_2 \end{array}$$

Si  $b$  est multiple de 4, on adopte aussi la notation hexadécimale.

Exemples

Taille	Entiers	Vecteurs binaires	Hexa
8	108	01101100	0x6c
8	27	00011011	0x3b
16	41668	1010001011000100	0xa2c4

# Incrémantation arithmétique

Incr :  $x = (x_{b-1}, \dots, x_1, x_0) \mapsto y = (y_{b-1}, \dots, y_1, y_0)$ , définie par

$$([y_{b-1} \dots y_1 y_0]_2) = ([x_{b-1} \dots x_1 x_0]_2 + 1) \bmod 2^b$$

aussi notée  $[y_{b-1} \dots y_1 y_0]_2 = [x_{b-1} \dots x_1 x_0]_2 \boxplus_{2^b} 1$

Exemple avec  $b = 8$

	Entier naturel	Bloc de bits	Hex
$x$	108	01101100	0x6c
$x + 1$	109	01101101	0x6d
$x + 2$	110	01101110	0x6e
$x$	254	11111110	0xfe
$x + 1$	255	11111111	0xff
$x + 2$	0	00000000	0x00

# Algorithme d'Euclide étendu

Aperçu : EEA *Extended Euclid Algorithm*

- Entrées : deux entiers naturels  $a, b$
- Sorties : trois entiers  $d, u, v$  tels que  $d = \text{pgcd}(a, b)$  et  $d = au + bv$
- Algo : par divisions euclidiennes successives

Complexité :  $\text{poly}(\log(\max(a, b)))$

Application : calcul de l'inverse modulaire

si  $\text{pgcd}(a, b) = 1$ , alors  $u = a^{-1} \bmod b$

Remarque : s'adapte point par point pour les polynômes

# Annexes

## 1 Mathématiques

- Généralités
- Deux résultats de probabilités
- Algèbre générale
- Polynômes
- Arithmétique des entiers
- Arithmétique modulaire

# Structure de $\mathbb{Z}/n\mathbb{Z}$ , $n = pq$ contexte cryptographique

$(\mathbb{Z}/n\mathbb{Z}, +, \times)$  est un *anneau commutatif*.

- il y a les éléments inversibles : les entiers de  $\{0 \dots n - 1\}$  qui sont premiers avec  $n$ . Leur ensemble est noté  $(\mathbb{Z}/n\mathbb{Z})^\times$ , et leur nombre est noté  $\phi(n)$ .
- il y a les éléments non nuls non inversibles : ils sont diviseurs de zéro (l'anneau est *non intègre*)

## Théorème d'Euler

Si  $x \in \mathbb{Z}/n\mathbb{Z}$ ,  $\text{pgcd}(x, n) = 1$ , alors  $x^{\phi(n)} = 1 \pmod{n}$

En contexte cryptographique ( $p$  et  $q$  très grands) : (presque) tout élément non nul est inversible.

# Le groupe multiplicatif $\mathbb{F}_p^*$

Pour  $p$  premier,  $(\mathbb{Z}/p\mathbb{Z}, +, \times)$  est un *corps*, noté  $\mathbb{F}_p$ , ou  $GF(p)$ .

- Tout élément non nul est inversible :  $\mathbb{F}_p^\times = \mathbb{F}_p^* = \mathbb{F}_p - \{0\}$
- $(\mathbb{F}_p^*, \times)$  est un *groupe cyclique* :

$$\exists g \in \mathbb{F}_p^* = \{g^0 = 1, g^1 = g, g^2, \dots, g^{p-2}\}.$$

Un tel  $g$  est un *générateur* de  $\mathbb{F}_p^*$ .

- Un générateur n'est pas unique : ce sont les  $g^k$  avec  $\text{pgcd}(k, p-1) = 1$ , où  $g$  est l'un d'entre eux. Il y en a donc  $\phi(p-1)$ .
- Petit théorème de Fermat (deux formulations équivalentes)

- (1)  $\forall x \in \mathbb{F}_p^*, x^{p-1} = 1$
- (2)  $\forall x \in \mathbb{F}_p, x^p = x$

# Le groupe multiplicatif $\mathbb{F}_p^*$

Exemple numérique :  $p = 19$ ;  $g = 2$

- on sait d'avance que  $2^{18} = 1 \bmod 19$  (petit théorème de Fermat)
- les puissances de 2, avec exposants de 0 à 17 décrivent tout  $\mathbb{F}_{19}^*$ .

$$\begin{aligned}\mathbb{F}_{19}^* &= \{2^0, 2^1, \dots, 2^{17}\} \\ &= \{1, 2, 4, 8, 16, 13, 7, 14, 9, 18, 17, 15, 11, 3, 6, 12, 5, 10\}\end{aligned}$$

(ce qui reprouve que 2 est générateur)

- Il y a  $\phi(18) = 6$  générateurs : 2, 13, 14, 15, 3, 10

# Annexes

## 1 Mathématiques

## 2 Algorithmique

- Tri, listes
- Calcul algébrique
- Factorisation, logarithme discret

# Ordres de grandeur des puissances de 2

Un millier / Un million / Un milliard	$2^{10} / 2^{20} / 2^{30}$
Nombre de secondes dans un jour / dans une année	$2^{16,4} / 2^{25}$
Nombre d'humains sur Terre	$2^{32,5}$
Âge de la Terre en années / en secondes	$2^{32} / 2^{57}$
Âge de l'univers en années / en secondes	$2^{34} / 2^{59}$
Nombre d'atomes dans la Terre	$2^{170}$
Nombre d'atomes dans le Soleil	$2^{190}$
Nombre d'atomes dans la galaxie	$2^{233}$
Nombre d'atomes dans l'univers	$2^{265}$

# Annexes

## 2 Algorithmique

- Tri, listes
- Calcul algébrique
- Factorisation, logarithme discret

## Quelques complexités classiques sur les listes

- tri d'une liste de  $n$  éléments :  $O(n \log_2(n))$
- recherche d'un élément dans une liste de  $n$  éléments
  - cas **non triée** :  $O(n)$
  - cas **triée** :  $O(\log_2(n))$
- insertion d'un élément dans une liste triée :  $O(\log_2(n))$
- fusion de deux listes triées de  $n_1$  et  $n_2$  éléments en une liste triée :  
 $O(n_1 + n_2)$
- recherche d'intersection entre deux listes  $\mathcal{L}_1$  et  $\mathcal{L}_2$  de  $n_1$  et  $n_2$  éléments
  - cas  $\mathcal{L}_1$  et  $\mathcal{L}_2$  **non triées** :  $O(n_1 n_2)$
  - cas  $\mathcal{L}_1$  **triée** et  $\mathcal{L}_2$  **non triée** :  $O(\log_2(n_1) n_2)$
  - cas  $\mathcal{L}_1$  et  $\mathcal{L}_2$  **triées** :  $O(n_1 + n_2)$

# Annexes

## 2 Algorithmique

- Tri, listes
- **Calcul algébrique**
- Factorisation, logarithme discret

# Exponentiation, calculs matriciels

- Calcul d'exponentiation  $a^n = \underbrace{a \times \cdots \times a}_{n \text{ fois}}$  :
  - $O(n)$  multiplications par la méthode naïve
  - $O(\log_2(n))$  multiplications par la méthode binaire
- Produit matrice  $n \times n$  - vecteur :  $O(n^2)$  multiplications élémentaires
- Produit de deux matrices  $n \times n$  :  $O(n^3)$
- Triangulation/forme échelon d'une matrice  $n \times n$  par pivot de Gauss :  $O(n^3)$
- Calcul du déterminant d'une matrice  $n \times n$  :  $O(n^3)$
- Existence et calcul de l'inverse d'une matrice  $n \times n$  :  $O(n^3)$
- Résolution d'un système linéaire  $n$  équations  $n$  inconnues :  $O(n^3)$

# Annexes

## 2 Algorithmique

- Tri, listes
- Calcul algébrique
- Factorisation, logarithme discret

# Algorithmes de factorisation et leur complexité

Type	Complexité heuristique
Crible d'Eratosthène	$O(\sqrt{n})$
Algorithme de Fermat	$O( \sqrt{p} - \sqrt{q} ^2)$
Méthode de Pollard	$O(B \log n / \log B)$ si $p - 1$ est $B$ -lisse
Crible quadratique	$O(\exp((1 + o(1))\sqrt{\log n \log \log n}))$
Crible algébrique	$O(\exp(1.92(\log n)^{1/3}(\log \log n)^{2/3}))$

# Algorithmes de calcul de logarithme discret dans $\mathbb{F}_p^*$

Type	Complexité heuristique
Générique	
Essai exhaustif	$O(p)$
Pas de bébé-pas de géant	$O(\sqrt{p})$
Spécifique $\mathbb{F}_p$	
Pohlig-Hellman	$O\left(\sum_{i=1}^s (e_i (\log p + \sqrt{p_i}))\right)$ où $p - 1 = p_1^{e_1} \dots p_s^{e_s}$
Calcul d'indice basique	$O\left(\exp(c\sqrt{\log p \log \log p})\right)$
Calcul d'indice avancé	$O\left(\exp(1.92(\log p)^{1/3}(\log \log p)^{2/3})\right)$