

Ch. 5 Introduction à la Cryptographie

UE Master 2 : Sécurité des Systèmes d'Information

Sébastien Duval



D'après vous ?

C'est quoi la crypto (-graphie, -logie) ?

Si même la question est
DIFFICILE 🤔!



Contenu

I. Introduction

II. Hachage

III. Cryptographie symétrique (clef privée)

IV. Cryptographie asymétrique (clef publique)

V. Signature

VI. Certificats électroniques

VII. Aller plus haut

VIII. Aller plus bas

IX. Aller plus loin

X. Résumé



Première partie I

Introduction



Contenu Partie I : Introduction

- 1 Grands axes
- 2 Principes et résultats fondamentaux
- 3 Sécurité inconditionnelle et calculatoire
- 4 Modèles de l'attaquante
- 5 Attaques génériques



Objectifs de sécurité

La **cryptologie** (“science du secret”) vise à obtenir de la *confidentialité* **prouvée**, et par extension est utilisée pour garantir l'**intégrité**.

Confidentialité $\Rightarrow$ ne permettre l'accès aux données qu'aux personnes autorisées.

Intégrité $\Rightarrow$ détecter toute modification des données non-autorisée.

Preuve $\Rightarrow$ Fortes garanties*.

Un critère de sécurité supplémentaires est traité par la cryptologie :

Authentification $\Rightarrow$ garantir qu'on parle à la bonne personne.

*. Techniquement pas toujours des preuves.





Conception et analyse

La **cryptologie** se sépare en deux axes antagonistes et complémentaires :

Cryptographie : conception de fonctions/protocoles cryptologiques.

Cryptanalyse : analyse des fonctions/protocoles cryptologiques :

- a priori** : preuves formelles durant la conception,

- a posteriori** : analyses statistiques en “boîte noire”, comme un attaquant.



D'après vous ?

La **cryptographie** est un domaine d'*optimisation*, de deux critères majeurs conjointement :

Sécurité – C'est quoi ? Comment ça se mesure ?

Coût – C'est quoi ? Comment ça se mesure ?

Si même la question est
DIFFICILE GfG!





Cryptographie

La **cryptographie** est un domaine d'*optimisation*, de deux critères majeurs conjointement :

Sécurité – mesurée en “**nombre de bits**” :

⇒ N “bits de sécurité” correspond à une complexité de 2^N :

Temps : complexité mesurée en nombre d'opérations élémentaires

Mémoire : complexité mesurée en nombre de bits à stocker

Données : complexité en nombre d'échantillons

Coût – en

🕒 €/\$

🕒 énergie

🕒 temps

🕒 taille de code / circuit

🕒 mémoire

Ces critères sont incompatibles : la sécurité implique d'utiliser des *fonctions pseudo-aléatoires*, mais *l'aléatoire coûte cher*. On doit donc trouver un compromis.





Contenu Partie I : Introduction

- 1 Grands axes
- 2 Principes et résultats fondamentaux
- 3 Sécurité inconditionnelle et calculatoire
- 4 Modèles de l'attaquante
- 5 Attaques génériques



Principe de Kerckhoffs

Principe de Kerckhoffs

Publié en 1883 dans "La Cryptographie Militaire" (cf. sur Arche, intéressant et clair)

La sécurité d'un cryptosystème ne doit reposer que sur le secret de la **clef**.

Clef

Une **clef** cryptographique est le pendant d'une clef de cadenas :

- 🔒 seules les personnes autorisées la possèdent (elle est secrète),
- 🔒 elle permet l'accès à une zone restrictive.

Jusqu'à la 1ère guerre mondiale, la méthode de chiffrement était le secret, porté sur le champ de bataille.

e.g. : 1ère guerre mondiale, au bout d'un mois, les français savaient déchiffrer tous les messages allemands.





Principe de Kerckhoffs

Principe de Kerckhoffs

Publié en 1883 dans “La Cryptographie Militaire” (cf. sur Arche, intéressant et clair)

La sécurité d'un cryptosystème ne doit reposer que sur le secret de la **clef**.

Clef

Une **clef** cryptographique est le pendant d'une clef de cadenas :

- seules les personnes autorisées la possèdent (elle est secrète),
- elle permet l'accès à une zone restrictive.

⇒ la sécurité ne doit pas reposer sur :

- le secret d'un identifiant
- le secret du cryptosystème utilisé
- un système offusqué (i.e. incompréhensible)





Vocabulaire du chiffrement

Un système cryptographique (**cryptosystème**) qui permet la Confidentialité est appelé **chiffrement**.

clef : le matériel secret du chiffrement, valeur **uniformément aléatoire**

clair : message d'entrée d'un chiffrement qu'on veut rendre confidentiel

chiffré : message de sortie du chiffrement, illisible pour quiconque ne connaît pas la clef

chiffrer : transformer le clair en chiffré

déchiffrer : transformer le chiffré en clair

"crypter" : ce mot n'existe pas

décrypter : transformer le chiffré en clair sans la clef

IV : Valeur initiale publique

Nonce : Valeur publique associée à une clef. Le cryptosystème ne doit jamais utiliser deux fois le même couple (clef, nonce)



D'après vous ?

Vous en connaissez, des chiffrements ?

Si même la question est
DIFFICILE GfGf!





Chiffrement de Vernam - One-Time-Pad (1)

Le chiffrement de Vernam est le **seul chiffrement parfait** [†]

Résultat fondateur de Claude Shannon rendu publique en 1949.

Chiffrement de Vernam

Si Alice veut chiffrer un message de n bits, il lui faut une clef secrète de n bits.

Soit k cette clef et m le clair, le chiffré est $c = m \oplus k$ (m xork).

Pour déchiffrer (connaissant k), il suffit de faire $m = c \oplus k$.

Alice ne doit plus jamais chiffrer avec cette clef k : la clef est **jetée** après usage.

[†]. À équivalence près.





Chiffrement de Vernam - One-Time-Pad (2)

Avantage

sécurité ultime

(petit exemple sur 3 lettres)

Inconvénients

si Alice veut envoyer un message à Bob, Bob doit connaître la clef.

i.e.

pour pouvoir partager secrètement un message m de n bits, Alice et Bob doivent déjà partager un message k de n bits.

Comment améliorer ?



concevoir un protocole efficace de partage de clef

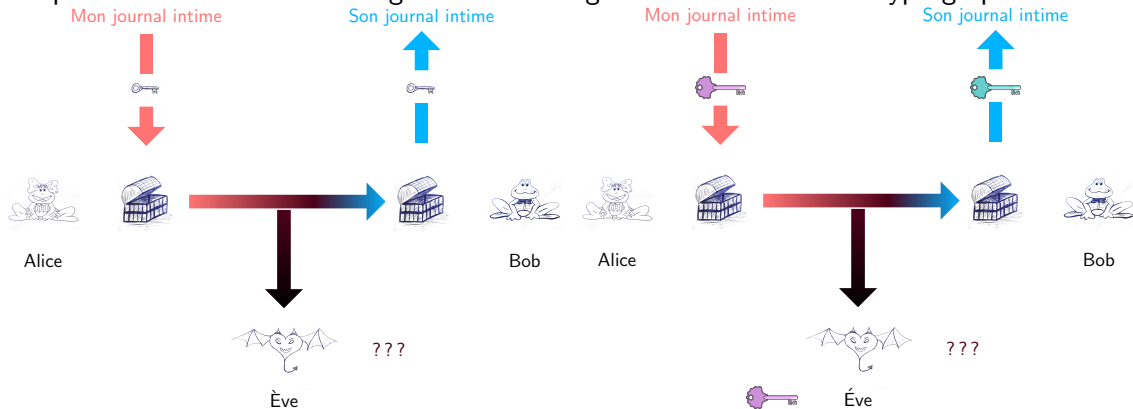


concevoir un cryptosystème avec une clef plus petite que le message



Cryptographies symétrique et asymétrique

Ces pistes d'amélioration ont engendré les deux grands domaines de la cryptographie :

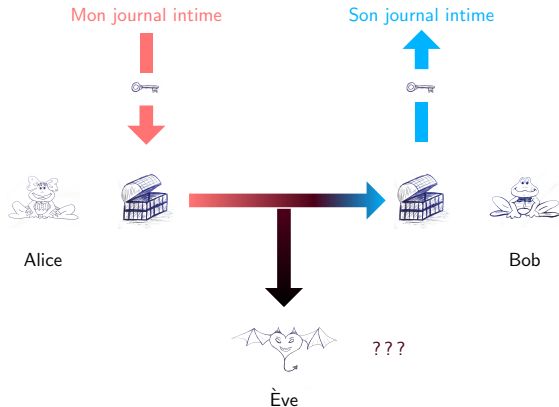


Cryptographie symétrique ("clef privée")

Cryptographie asymétrique ("clef publique")



Cryptographie symétrique



Alice et Bob **partagent la même clef k** .

- 🧠 spécialisé avec $|k| \ll |m|$
($\approx |m| > 2^{60}|k|$)
- 🧠 à coût très faible
- 🧠 aussi vieux que le langage

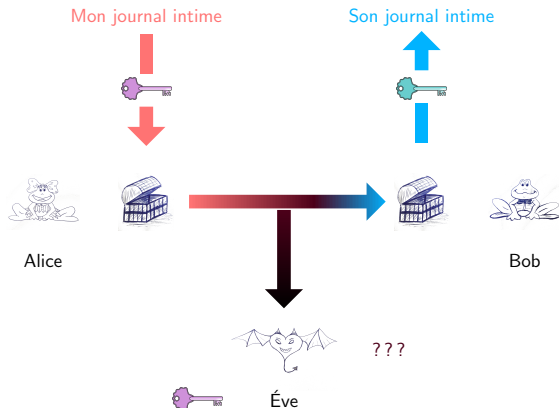
Cryptographie symétrique ("clef privée")

Cryptographie symétrique ("clef privée")





Cryptographie asymétrique



clef privée : clef **connue uniquement par Bob**, sert à **déchiffrer**,
clef publique : clef **connue de tous**, sert à **chiffrer**.

- 🧠 spécialisé dans le **partage de clef**
- 🧠 coût élevé ($|k| > |m|$)
- 🧠 inventé dans les années 1970

Cryptographie asymétrique ("clef publique")

Cryptographie asymétrique ("clef publique")





Chiffrement hybride

Rappel : les 2 inconvénients du OTP

🧐 $|k| = |m|$ $\Rightarrow$ résolu par le chiffrement symétrique

🧐 Alice et Bob doivent partager une clef secrète $\Rightarrow$ résolu par le chiffrement asymétrique

$\Rightarrow$ **on combine les deux !**

1. Bob crée un couple (k_{priv}, k_{pub})

2. Alice crée une clef k_{sym}

3. Alice partage k_{sym} chiffrée avec k_{pub} (i.e. asymétrique avec $m = k_{sym}$)

4. Alice et Bob partagent désormais un secret, k_{sym} , de n bits

5. Alice et Bob communiquent jusqu'à $2^{60}|k|$ bits de clair

6. Avant d'atteindre la limite, Alice et Bob se communiquent une nouvelle clef k'_{sym}





Canal symétrique

Alice

Bob

Échanges sécurisés

$Enc(\text{"Message secret"}, K_{\text{sym}}) = \text{"oijjaziurfouzeo"}$

"oijjaziurfouzeo"
(chiffré + authentifié)

$Dec(\text{"oijjaziurfouzeo"}, K_{\text{sym}}) = \text{"Message secret"}$

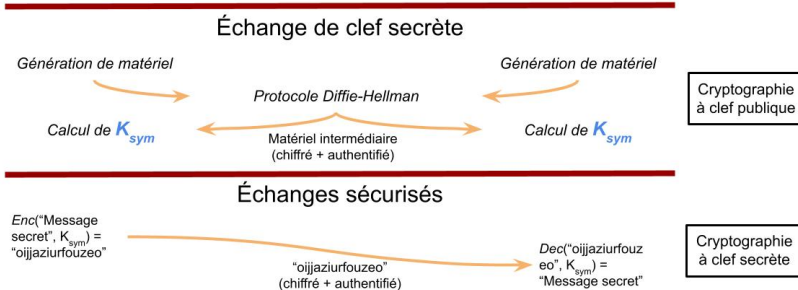
Cryptographie
à clef secrète



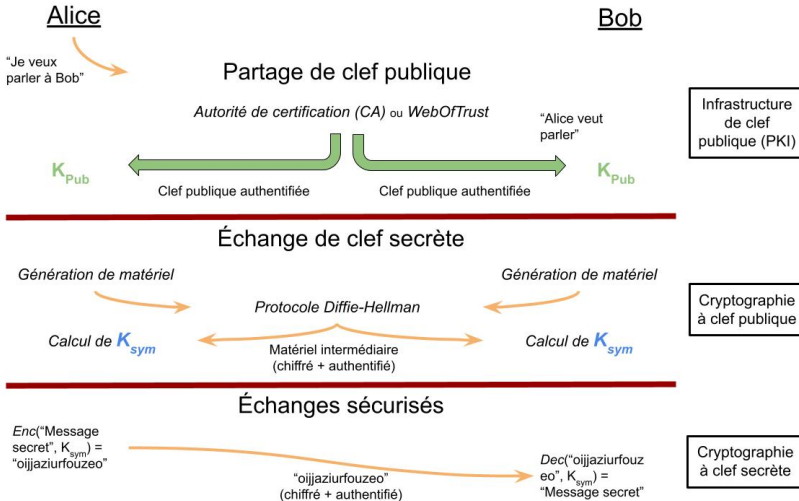
Canal asymétrique puis symétrique

Alice

Bob

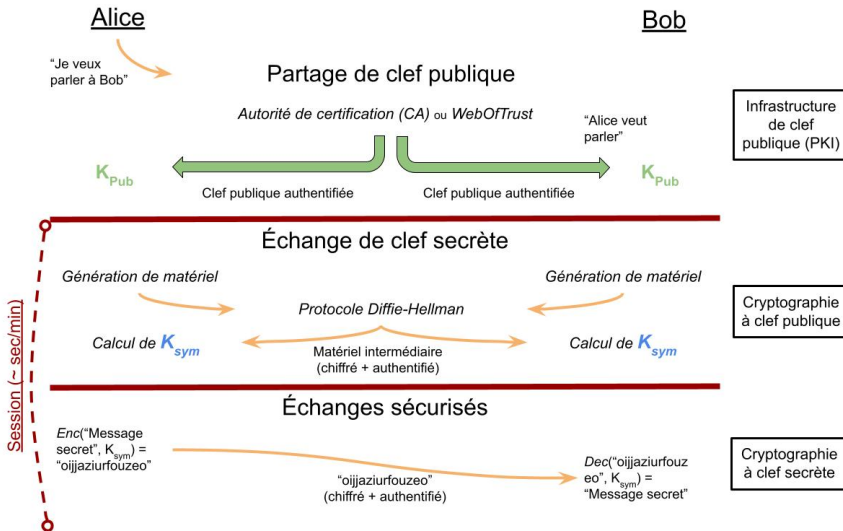


Canal hybride





Canal hybride avec session de quelques secondes/minutes





Canal sécurisé

Note : *c'est une version approximative pour l'introduction*
Schéma complet en Section Résumé.



Contenu Partie I : Introduction

- 1 Grands axes
- 2 Principes et résultats fondamentaux
- 3 Sécurité inconditionnelle et calculatoire**
- 4 Modèles de l'attaquante
- 5 Attaques génériques



Sécurité inconditionnelle

Rappel : Shannon a prouvé que seul le OTP (chiffrement de Vernam) a une sécurité parfaite. Mais il est difficile à utiliser en pratique.

La sécurité est-elle possible ??

La réponse est oui.

Sécurité inconditionnelle

Il n'existe pas assez d'information pour pouvoir casser la sécurité.

Shannon : le OTP est le seul chiffrement inconditionnellement sûr.

Note : c'est de la sécurité contre un adversaire **illimité**.

Mais aucun adversaire n'est illimité !

Dès qu'on impose une limite (même inatteignable), il n'y a plus de preuve que la sécurité est impossible.





Sécurité calculatoire

Définition 1

Sécurité calculatoire : Notion de sécurité “réaliste”.

Casser la sécurité coûte plus cher que les moyens de l'attaquant.

D'où la mesure en “**bits de sécurité**”.

Sécurité actuelle

Excellente : 128 bits (résiste à tout attaquant)

Suffisante : 64 bits (résiste à tout sauf NSA, Google et quelques états)



D'après vous ?

128 bits, c'est beaucoup ?

Si même la question est
DIFFICILE GfG!





Contenu Partie I : Introduction

- 1 Grands axes
- 2 Principes et résultats fondamentaux
- 3 Sécurité inconditionnelle et calculatoire
- 4 Modèles de l'attaquante
- 5 Attaques génériques



Définition

Que connaît l'attaquante ?

- 🧐 le fonctionnement du cryptosystème (principe de Kerckhoffs)
- 🧐 p messages clairs ?
- 🧐 q messages chiffrés ?
- 🧐 ℓ couples (clair, chiffré) ?

Que veut trouver l'attaquante ?

- 🧐 le clair associé à un message chiffré
- ou
- 🧐 (plus fort) la clef secrète

Que choisit l'attaquante ?

- 🧐 les clairs ?
- 🧐 les chiffrés ?



Boîte noire

Principe : par défaut, on considère l'attaquante la plus forte car on veut lui résister.

Boîte noire

- 🧐 Ève ne connaît pas :
 - 🧐 la clef secrète
- 🧐 Clair connu (Known-plaintext) :
Ève connaît mais ne choisit pas les entrées
- 🧐 Chiffré connu (Known-ciphertext) :
Ève connaît mais ne choisit pas les sorties
- 🧐 Clair choisi (Chosen-plaintext) :
Ève les entrées
- 🧐 Chiffré choisi (Chosen-ciphertext) :
Ève choisit les sorties





Exemples de modèles avancés

Modèles avancés

- 🧑 Clef-liée (Related-key) :
Ève connaît/choisit des entrées/sorties pour 2 clefs k_1 et k_2 et connaît $k_1 \oplus k_2$
- 🧑 IV/Nonce, ... choisi

Boîte grise

Un modèle réaliste donne à Ève la possibilité de faire des mesures physiques sur le cryptosystème, ou d'exploiter des faiblesses du système d'exploitation.

Voir "Pour aller plus bas – Attaques par canaux auxiliaires".

Boîte blanche

Un modèle (souvent trop) fort donne à Ève la connaissance voire le choix de la clef.
Intéressant pour l'analyse, parfois réaliste en pratique.

Voir "Pour aller plus haut – Boîte blanche".





Contenu Partie I : Introduction

- 1 Grands axes
- 2 Principes et résultats fondamentaux
- 3 Sécurité inconditionnelle et calculatoire
- 4 Modèles de l'attaquante
- 5 **Attaques génériques**



Attaques génériques - force brute

Attaque par force brute

Une attaque fonctionne toujours : soit k une clef de N bits.

L'algorithme suivant renvoie la valeur de k .

```
obtenir un couple  $(m, E_k(m))$ 
pour tout  $x$  dans  $\{0, \dots, 2^N\}$  faire
    si  $E_x(m) = E_k(m)$  alors
        retourner  $x$ 
    fsi
fpour
```

Coût : $2^N \times \text{Coût}(E_k)$

⇒ **pour une sécurité de N bits, il faut une clef d'au moins N bits !**



Attaques génériques - distingueur

Définition 2

Distingueur : e.g. force brute : distinguer la bonne clef par le test $E_x(m) = E_k(m)$.

Ce test est un **distingueur** :

- mauvaise clef $\rightarrow$ comportement aléatoire
- bonne clef $\rightarrow$ comportement prédictible

Toute **propriété non-aléatoire** de E_k , observable en coût 2^M , est un distingueur de coût 2^M .

Distingueur et attaque

Distinguer le cryptosystème d'une fonction aléatoire n'est pas une attaque.

Cependant, *un distingueur conduit presque toujours à une attaque* avec un peu plus de travail

$\Rightarrow$ *un cryptosystème avec un distingueur peu coûteux ne doit pas être utilisé*.



Attaques génériques - collision

Collision

Soit f une fonction.

Une **collision** est l'obtention de $x \neq x'$ tels que $f(x) = f(x')$.

En cryptographie, obtenir une collision mène souvent à une attaque.

Coût de collision pour une fonction uniforme

Avec $f : A \rightarrow B$ uniforme, **après** $\sqrt{\#B}$ **tirages** dans A , la probabilité de **collision** est $\geq 1/2$.

C'est la **borne des anniversaires** (la fonction [personne $\mapsto$ anniversaire] a 365 dates de sortie : sur $\sqrt{365} = 19$ personnes, 2 ont probablement le même anniversaire)

Algorithmes : des algorithmes efficaces trouvent une collision, e.g. **rhô de Pollard**.

Si la sortie de f fait M bits, **collision en** $2^{M/2}$ **opérations !**

→ **pour une sécurité de N bits, préférer des fonctions sur au moins $2N$ bits !**



Deuxième partie II

Hachage



Contenu Partie II : Hachage

6 Fonction de hachage cryptographique

7 Applications

8 Standards

D'après vous ?

Vous connaissez des fonctions de hachage ? Cryptographiques ?

Si même la question est
DIFFICILE GfG!





Fonction de hachage cryptographique

Objectif

Outil pour l'**intégrité**.

Principe : fonction de $\{0, 1\}^* \rightarrow \{0, 1\}^n$

Hacher $\simeq$ calculer une **empreinte** cryptographique

Caractéristiques

Résistance aux collisions : impossible de trouver $m \neq m'$ tq $h(m) = h(m')$

Résistance aux premières préimages : connaissant d , impossible de trouver m tq $h(m) = d$

Résistance aux secondes préimages : connaissant m , trouver $m' \neq m$ tq $h(m) = h(m')$





Contenu Partie II : Hachage

6 Fonction de hachage cryptographique

7 Applications

8 Standards



Applications

- 🧐 **compresser** de grandes quantités de données pour la **signature** (pour ne pas signer un gros objet, et pour ajouter de l'indépendance entre les signatures)
- 🧐 assurer l'**intégrité**



Contenu Partie II : Hachage

6 Fonction de hachage cryptographique

7 Applications

8 Standards



Standards de fonctions de hachage

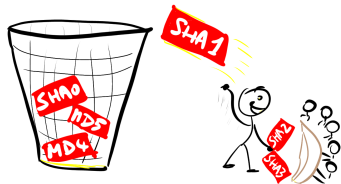
Obsolètes

- MD4 (Rivest 1990, collisions en 1995, préimages en 2005)
- MD5 (Rivest 1991, collisions en 2004)
- SHA-0 (NIST 1993, collisions en 1998)
- SHA-1 (NIST 1993, obsolète depuis 2005, collisions pratiques en 2020)

Bons choix

- SHA-256/224, SHA-512/384
- SHA-3 (Keccak), un peu cher

Q : Pourquoi SHA sur 256 et 512 bits ?





Troisième partie III

Chiffrement symétrique (clef privée)

Contenu Partie III : Chiffrement symétrique (clef privée)

- 9 Objectif
- 10 Chiffrement à flot (stream cipher)
- 11 Chiffrement par blocs
- 12 Modes de chiffrement par blocs
- 13 Authentification
- 14 Chiffrement authentifié
- 15 Résumé



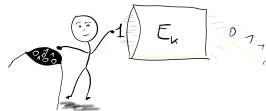
Objectif

Sachant qu'Alice et Bob connaissent un secret k de 64/80/128/256 bits, leur permettre d'échanger confidentiellement $\approx 2^{|k|}$ bits d'information.



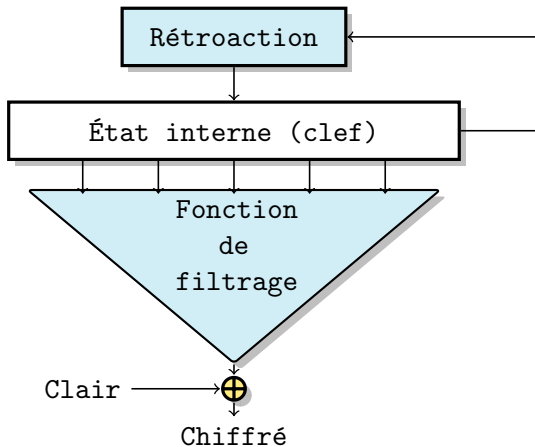
Contenu Partie III : Chiffrement symétrique (clef privée)

- 9 Objectif
- 10 Chiffrement à flot (stream cipher)
- 11 Chiffrement par blocs
- 12 Modes de chiffrement par blocs
- 13 Authentification
- 14 Chiffrement authentifié
- 15 Résumé





LFSR



Un **LFSR** (registre à décalage à rétroaction linéaire) est composé de :

état interne : initialisé avec la clef, décalé de 1 bit à chaque coup d'horloge

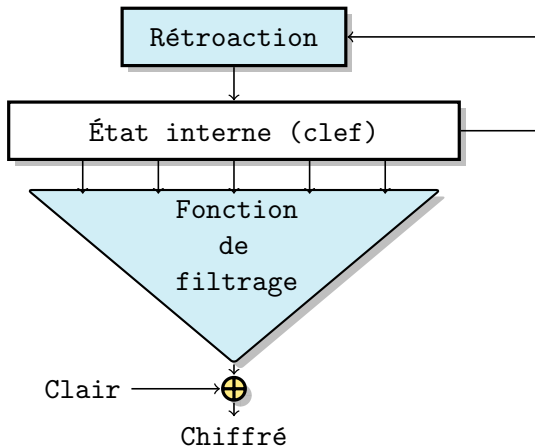
rétroaction : génère un bit à chaque coup d'horloge, qui est injecté dans l'état interne à l'emplacement laissé vide par le décalage

Dans un LFSR, la rétroaction est *linéaire* ($\approx$ peu complexe).

Il existe l'alternative du NLFSR avec rétroaction *non-linéaire*.



LFSR



La **fonction de filtrage** est une fonction Booléenne : elle prend en entrée tout l'état interne et génère 1 bit à chaque coup d'horloge. C'est une fonction non-linéaire.

Le bit généré est xorié à un bit de clair pour donner un bit de chiffré, ou à un bit de chiffré pour déchiffrer un bit de clair.



Standards

Quelques standards.

🧐 Vernam (techniquement)

🧐 orientés logiciel (software) :

Salsa20 rapide en software mais gros

ChaCha Salsa20 en mieux

🧐 orientés matériel (hardware) :

Trivium petit, rapide, basse énergie

Kreyvium Trivium pour applications spécifiques

Note : des chiffrements obsolètes (A5/1, 15/2, RC4, WAKE, ...) sont encore très utilisés.

Ailleurs : même hors de la cryptographie, les chiffrements à flot sont très utilisés pour générer du pseudo-aléa peu coûteux.

Contenu Partie III : Chiffrement symétrique (clef privée)

- 9 Objectif
- 10 Chiffrement à flot (stream cipher)
- 11 Chiffrement par blocs**
- 12 Modes de chiffrement par blocs
- 13 Authentification
- 14 Chiffrement authentifié
- 15 Résumé



Principe

Chiffrer/déchiffrer les messages par blocs de 64/80/128/256 bits.

Intérêts :

- 👤 simple d'utilisation
- 👤 efficace sur de gros processeurs
- 👤 meilleure sécurité que le chiffrement à flot

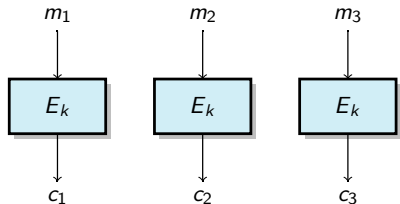
Inconvénient :

- 👤 un peu plus coûteux

...

!! l'appellation **chiffrement par blocs** est souvent donnée^a à la fonction E_k . Mais E_k n'est pas un chiffrement ! (un chiffrement a une entrée de taille quelconque)

^a. À tort mais c'est comme ça.



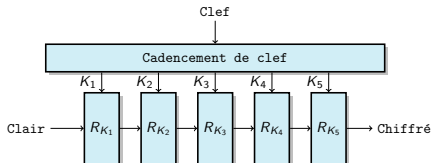


Chiffrement itéré

E_k

Idéalement, E_k est une **famille de**

permutations (bijections) aléatoires.



D'après vous ?

Comment ça "aléatoire" ? C'est facile ça "aléatoire" ?

Comment on fait pour tirer un nombre "au hasard" ?

Si même la question est
DIFFICILE *Geog!*

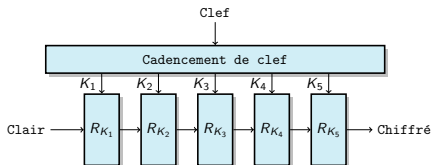


Chiffrement itéré

Idéalement, E_k est une **famille de**

E_k

permutations (bijections) aléatoires.



Mais l'aléa coûte cher à implémenter : on met de la **structure** dans E_k .

E_k est définie comme l'**itération** (la composition) d'une **fonction de tour** R , paramétrée à chaque tour par une clef secrète K_r .

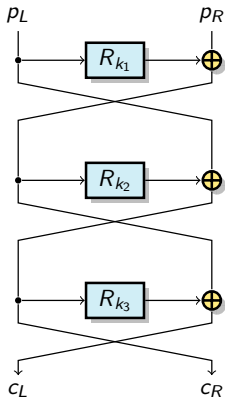
R est une fonction **simple**, **peu coûteuse**, **donnant peu de sécurité**.

Deux familles majeures de E_k : Feistel et SPN.



Feistel

du nom de Horst Feistel, créateur du premier standard cryptographique DES.



On réduit la sécurité et le coût de E_k à ceux de R , sur $N/2$ bits

Avantages :

- 🔍 simplification du problème
- 🔍 E_k^{-1} est E_k est inverse l'ordre des clefs de tour $(k)_{1,\dots,r}$.

Exemples :

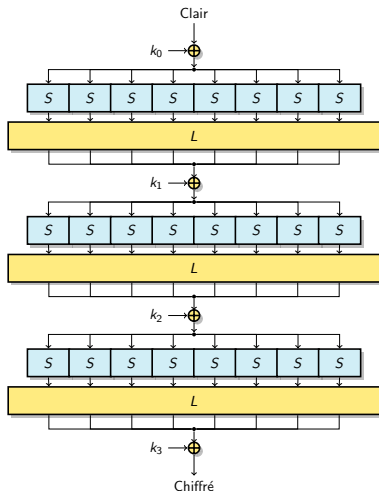
DES (Data Encryption Standard), obsolète
(clef trop petite aujourd'hui)

LBlock très apprécié en Chine

Blowfish/Twofish dans HTTPS



Substitution-Permutation Network



On réduit la sécurité et le coût de E_k à ceux de

- S non-linéaire sur 3, 5, 8 bits
- L linéaire sur N , $N/2$ ou $N/4$ bits

Avantages :

- réduction à des composants simples

Exemples :

AES (Advanced Encryption Standard),
LE standard symétrique

PRESENT standard à bas coût (un peu vieux)

Skinny standard ISO à bas coût

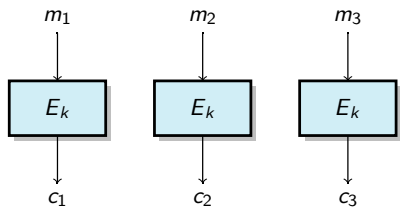
Ascon tout nouveau standard NIST à bas coût

Contenu Partie III : Chiffrement symétrique (clef privée)

- 9 Objectif
- 10 Chiffrement à flot (stream cipher)
- 11 Chiffrement par blocs
- 12 Modes de chiffrement par blocs**
- 13 Authentification
- 14 Chiffrement authentifié
- 15 Résumé



Principe



Mode ECB (Electronic CodeBook)

Problèmes :

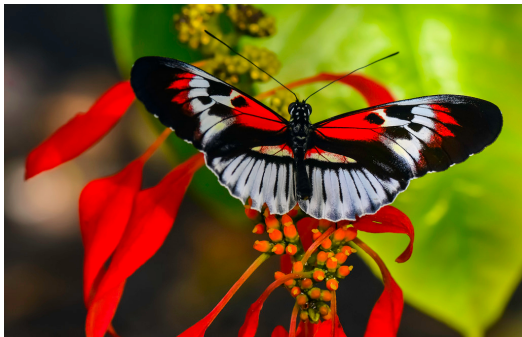
- 🔒 Un chiffrement par blocs n'est *pas un chiffrement*.
- 🔒 Appliquer E_k en parallèle (mode ECB) est un *mauvais* chiffrement.

Q : Comment obtenir un bon chiffrement à partir d'un ..chiffrement par blocs ?

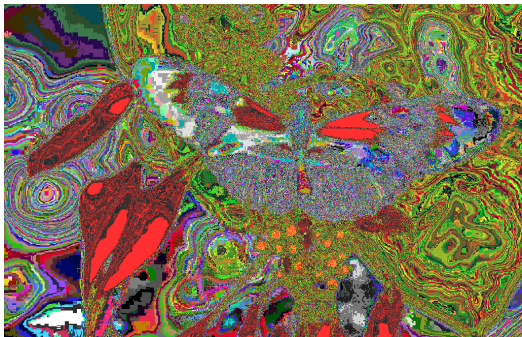
On utilise un **mode opératoire**.



Exemple d'un mauvais mode opératoire (ECB).

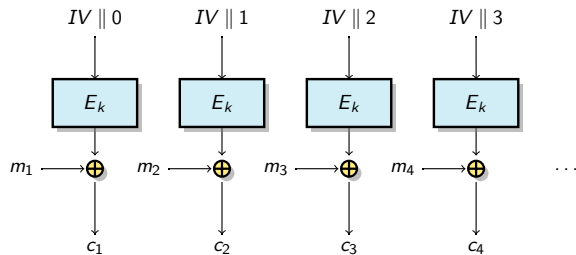
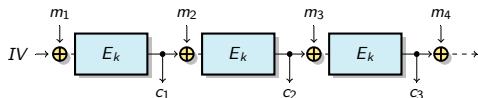


Une image anodine d'un papillon.



Après ECB : on voit le papillon.

Standards



Mode CBC : chaînage simple

Inconvénient : latence

Mode CTR : compteur

Avantage : parallèle

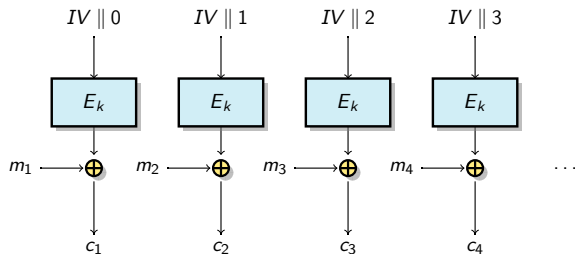
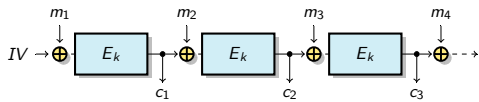
Note : c'est un chiffrement à flot par blocs !

🔒 chiffrement = déchiffrement !!

🔒 si IV n'est pas un nonce, c'est cassé !



Standards



Mode CBC : chaînage simple

Mode CTR : compteur

Inconvénient : sur ces modes, collision $\rightarrow$ attaque (cf. Sweet32 sur Arche)

$\Rightarrow$ sécurité réduite à $B/2$ bits, avec B la taille de bloc !

🧐 $E_k = \text{DES ou 3DES (blocs de 64 bits)} \rightarrow$ **32 bits de sécurité !!** (accepté par 19% des sites)

🧐 $E_k = \text{AES (blocs de 128 bits)} \rightarrow$ **64 bits de sécurité !!** (accepté par 66% des sites)

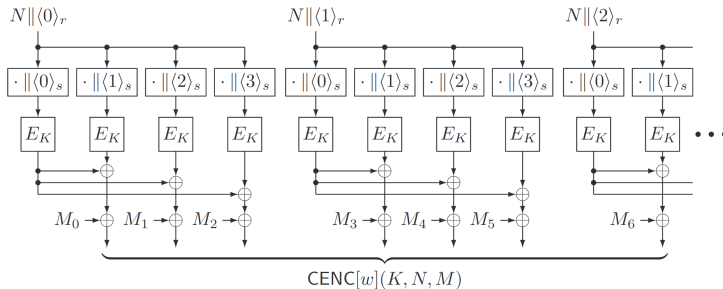


Modes résistants aux collisions

Il existe des modes résistants aux collisions (sécurité $> B/2$ bits).

Ils sont plus complexes et légèrement plus coûteux (et pas utilisés).

Exemples : CENC, PMAC+



Mode CENC

Contenu Partie III : Chiffrement symétrique (clef privée)

- 9 Objectif
- 10 Chiffrement à flot (stream cipher)
- 11 Chiffrement par blocs
- 12 Modes de chiffrement par blocs
- 13 **Authentification**
- 14 Chiffrement authentifié
- 15 Résumé

Principe

Authentification

Vérifier l'identité de l'expéditeur.

(indirectement, assurer l'**intégrité** des données (non-modification)).

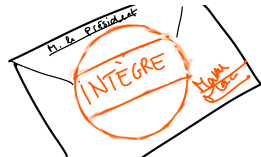
Cryptosystème : **algorithme de MAC** (Message Authentication Code).

MAC

MAC ou **étiquette** : une donnée courte, ajoutée au message, qui garantit son authenticité.

La plupart des algorithmes de MAC sont construits comme une combinaison :

hachage + chiffrement par blocs





Modes

Exemples de modes d'authentification :

CBC-MAC : standard FIPS-113 (le MAC est le chiffré de la sortie du mode CBC)

OMAC : CBC-MAC en mieux

HMAC : itération de 2 hachages cryptographiques (hachage de [clef || message])

Siphash, Chaskey : ad-hoc (construits comme des chiffrements par blocs)

Hash-then-PRF : généralement sous la forme “Hachage puis chiffrement symétrique du haché”

BB : les modes de MAC sont particulièrement sensibles aux **collisions** !

Modes résistants aux collisions

Tout algorithme de MAC déterministe est limité à la borne des anniversaires.

(MAC de B bits : il faut $2^{B/2}$ essais avant de trouver deux MACs égaux pour des messages différents)

⇒ on rend les MAC non-déterministes avec un **nonce** N .

Exemples de modes d'authentification résistants aux collisions :

Wegman-Carter, EWCDM, Hash-then-PRF avec nonce.

Bref : c'est possible, avec des nonces.

Contenu Partie III : Chiffrement symétrique (clef privée)

- 9 Objectif
- 10 Chiffrement à flot (stream cipher)
- 11 Chiffrement par blocs
- 12 Modes de chiffrement par blocs
- 13 Authentification
- 14 Chiffrement authentifié**
- 15 Résumé



Chiffrement authentifié

Pour simplifier l'utilisation, on a des modes qui font (a) chiffrement et (b) authentification.

Exemples : CCM (CTR+CBC-MAC), GCM (CTR+Hash-then-PRF avec nonce), OCB

TLS : on authentifie et chiffre chaque paquet → on utilise de plus en plus des modes de chiffrement authentifié.

Contenu Partie III : Chiffrement symétrique (clef privée)

- 9 Objectif
- 10 Chiffrement à flot (stream cipher)
- 11 Chiffrement par blocs
- 12 Modes de chiffrement par blocs
- 13 Authentification
- 14 Chiffrement authentifié
- 15 **Résumé**

Résumé

La cryptographie symétrique, c'est :

- le chiffrement avec clef partagée
- l'authentification avec clef partagée
- le chiffrement authentifié avec clef partagée

→ confidentialité
→ authenticité
→ les deux

Les blocs constituant sont :

- des fonctions de hachage cryptographiques
- des chiffrements par blocs (Feistel, SPN)
- des modes opératoires (de chiffrement et d'authentification)

Assure :

- bas coût
- petite clef (128 bits), grands messages ($\approx 2^{60}$ bits)





Quatrième partie IV

Chiffrement asymétrique (clef publique)



Contenu Partie IV : Chiffrement asymétrique (clef publique)

16 Introduction

17 Quelques problèmes difficiles utilisés en cryptographie

18 Standards

D'après vous ?

Rappelez-moi, c'est quoi la cryptographie asymétrique ?

Si même la question est
DIFFICILE GfG!





Fonction à sens unique et porte dérobée

en anglais : One-way function & trapdoor

Définition

Une **fonction à sens unique** est :

- 🧐 facile à calculer
- 🧐 difficile à inverser

Une **porte dérobée** est une **information secrète** qui rend l'inversion facile

→ ça paraît bien pour la cryptographie !

On n'a qu'à dire que **la porte dérobée est la clef secrète**
(de déchiffrement) !





Possible ?

La sécurité à clef publique est-elle possible ?

Être ou ne pas être sûr...





Possible ?

La sécurité à clef publique est-elle possible ?

Réponses :

🧐 “Oui, si vous posez la question et qu’il y a un cours dessus. . . ”

Être ou ne pas être sûr...





Possible ?

La sécurité à clef publique est-elle possible ?

Réponses :

- 🗣️ “Oui, si vous posez la question et qu’il y a un cours dessus. . . ”
- 🗣️ “Oui, je connais des algorithmes à clef publique, RSA par exemple !”

Être ou ne pas être sûr...





Possible ?

La sécurité à clef publique est-elle possible ?

Réponses :

- 🗣️ “Oui, si vous posez la question et qu’il y a un cours dessus. . . ”
- 🗣️ “Oui, je connais des algorithmes à clef publique, RSA par exemple !”

La vraie réponse est : **on ne sait pas, mais on le suppose.**

La cryptographie asymétrique se construit généralement sur des problèmes **réputés difficiles** (parfois prouvés NP-difficiles).

Mais il pourrait y avoir des solutions triviales, on ne sait pas.

Être ou ne pas être sûr...





Difficile

La question fondamentale à se poser est : **“Qu’est-ce qui est difficile ?”**

Si même la question est
DIFFICILE *Geog!*





Difficile

La question fondamentale à se poser est : **“Qu’est-ce qui est difficile ?”**

🧐 La réponse de Shannon et de la cryptographie symétrique est :

“Difficile, c’est trop coûteux pour l’attaquante ; c’est la sécurité calculatoire !”

Si même la question est
DIFFICILE 🤔!





Difficile

La question fondamentale à se poser est : **“Qu’est-ce qui est difficile ?”**

🗣️ La réponse de Shannon et de la cryptographie symétrique est :

“Difficile, c’est trop coûteux pour l’attaquante ; c’est la sécurité calculatoire !”

🗣️ La réponse en cryptographie à clef publique est :

“Mais le coût c’est dur à estimer, et ça évolue avec les technologies. Est-ce qu’on ne peut pas avoir une **définition mathématique formelle** plutôt ?”

Si même la question est
DIFFICILE 🤔!





Difficile

La question fondamentale à se poser est : **“Qu’est-ce qui est difficile ?”**

🗣️ La réponse de Shannon et de la cryptographie symétrique est :

“Difficile, c’est trop coûteux pour l’attaquante ; c’est la sécurité calculatoire !”

🗣️ La réponse en cryptographie à clef publique est :

“Mais le coût c’est dur à estimer, et ça évolue avec les technologies. Est-ce qu’on ne peut pas avoir une **définition mathématique formelle** plutôt ?”

La réponse est **non**.

Mais les cryptographes n’ont pas voulu l’entendre, donc ils l’ont fait quand même !

Problème : leur définition de “difficulté” mathématique ne correspond ni au coût d’attaque, ni à la sécurité...

Si même la question est
DIFFICILE 🤔!



D'après vous ?

Ça vous parle, la complexité algorithmique ?

Si même la question est
DIFFICILE GfG!





Difficulté et sécurité

La définition mathématique formelle en cryptographie à clef publique est :

“Difficile, c’est au sens de la **complexité asymptotique** ; c’est **NP-difficile** !”



Difficulté et sécurité

La définition mathématique formelle en cryptographie à clef publique est :

“Difficile, c’est au sens de la **complexité asymptotique** ; c’est **NP-difficile** !”

D’après vous ?

NP-difficile, c’est très difficile ?



Difficulté et sécurité

La définition mathématique formelle en cryptographie à clef publique est :

“Difficile, c’est au sens de la **complexité asymptotique** ; c’est **NP-difficile** !”

D’après vous ?

NP-difficile, c’est très difficile ?

- 🧐 le parcours optimal de tous les sommets d’un graphe, c’est NP-difficile. . . pourtant votre facteur le fait tous les jours. . .
- 🧐 programmer optimalement un algorithme donné, c’est NP-difficile. . . pourtant l’informatique, ça marche. . .
- 🧐 gagner à Mario, c’est NP-difficile. . . pourtant vous y arrivez (parfois même sans tricher) !



Difficulté et sécurité

La définition mathématique formelle en cryptographie à clef publique est :

“Difficile, c’est au sens de la **complexité asymptotique** ; c’est **NP-difficile** !”

Inconvénients :

- 🧐 complexité asymptotique (clef infiniment grande) : ne dit rien pour des tailles réalistes



Difficulté et sécurité

La définition mathématique formelle en cryptographie à clef publique est :

“Difficile, c’est au sens de la **complexité asymptotique** ; c’est **NP-difficile** !”

Inconvénients :

- 🧐 complexité asymptotique (clef infiniment grande) : ne dit rien pour des tailles réalistes
- 🧐 NP-difficile veut dire qu’il existe au moins une clef difficile à attaquer
Mais avec la clef qu’on a : pas de garanties.



Difficulté et sécurité

La définition mathématique formelle en cryptographie à clef publique est :

“Difficile, c’est au sens de la **complexité asymptotique** ; c’est **NP-difficile** !”

Inconvénients :

- 🧐 complexité asymptotique (clef infiniment grande) : ne dit rien pour des tailles réalistes
- 🧐 NP-difficile veut dire qu’il existe au moins une clef difficile à attaquer
Mais avec la clef qu’on a : pas de garanties.

Conséquence : NP-difficile n’est qu’un guide de conception

- 1️⃣ chercher une fonction à sens unique, où l’inversion est NP-difficile sans clef
- 2️⃣ vérifier la sécurité (difficile au sens coût)



Contenu Partie IV : Chiffrement asymétrique (clef publique)

16 Introduction

17 Quelques problèmes difficiles utilisés en cryptographie

18 Standards



Factorisation (dans un corps fini)

*Note : la factorisation n'est **pas** NP-difficile.*

Factorisation

p, q entiers, premiers.

- 🧐 $(p, q) \rightarrow n = p.q$ facile (quadratique en le nb de bits de p et q)
- 🧐 $n = p.q \rightarrow (p, q)$ difficile (à notre connaissance), mais pas NP-difficile

J'ai trouvé les facteurs!!





RSA

D'où le grand classique RSA (Rivest, Shamir, Adelman)

RSA

On donne $n = pq$, avec p et q des premiers.

Clef publique : (e, n) , [e aléatoire avec $\text{pgcd}(e, \phi(n)) = 1$ (où $\phi(n) = \text{ppcm}(p-1, q-1)$)]

Clef secrète : $d = e^{-1} \bmod \phi(n)$

Chiffrement : $c = m^e \bmod n$

Déchiffrement : $m = c^d \bmod n$

On a :

$$c^d = m^{de} = m.m^{k\phi(n)} = m \bmod n$$

D'après le petit théorème de Fermat : $\forall x \in (\mathbb{Z}/n\mathbb{Z}), x^{\phi(n)} = 1$.



Autres exemples

RSA	factorisation d'entiers
Rabin	factorisation d'entiers + racines carrées modulaires
ElGamal	logarithme discret
McEliece	décodage de code linéaire
Merkle-Hellman	somme de sous-ensembles
Goldwasser-Micali	résidus quadratiques

Grandes directions actuelles :

réseaux (lattices) : plus court vecteur, apprendre avec des erreurs, ...

courbes elliptiques : logarithme discret sur les courbes elliptiques (obligatoire dans TLS v1.3)



Contenu Partie IV : Chiffrement asymétrique (clef publique)

16 Introduction

17 Quelques problèmes difficiles utilisés en cryptographie

18 Standards



Standards

La standard de chiffrement à clef publique actuel est **RSA** avec une clef d'au moins 2048 bits. ‡

Sur les mêmes problèmes, on construit :

des signatures : standards DSA, ECDSA, EdDSA, ...

des échanges de clef : Diffie-Hellman, ECDH, ...

Note : le plus important, c'est les échanges de clef. Le chiffrement, on le fait plus efficacement en symétrique, et les signatures, on fait presque aussi bien plus efficacement en symétrique.

Le plus important, c'est d'échanger un secret initial pour pouvoir faire de la cryptographie symétrique

2 puissance...2048?



‡. La factorisation est possible en pratique pour $n = pq$, n faisant 829 bits.



Cinquième partie V

Signature



Contenu Partie V : Signature

19 Introduction

20 Signature à clef privée

21 Signature à clef publique



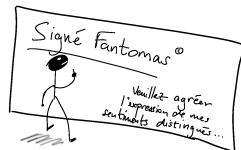
Objectif

Les garanties qu'apporte une **signature électronique** sont :

Intégrité : les données n'ont pas été modifiées

Authenticité : l'expéditeur est bien celui annoncé
et parfois

Non-répudiation : l'expéditeur ne peut nier avoir signé les données





Principe général – hachage

Avant de signer un message, on le hache !

⇒ avec un hachage cryptographique :

- 🗣️ on ajoute de la sécurité,
- 🗣️ la signature est moins coûteuse



Principe général – hachage

- 1 Alice hache $m \rightarrow h$
- 2 Alice signe h avec sa clef *privée* $k \rightarrow S_k(h)$
- 3 Alice envoie $(m, S_k(h))$ à Bob
- 4 Bob hache $m \rightarrow h$
- 5 Bob vérifie $S_k(h)$ avec l'algorithme $Verif_e(x)$, paramétré par la clef e §
- 6 Si $Verif_e(S_k(h)) = Vrai$, m est authentique et vient d'Alice
- 7 Si $Verif_e(S_k(h)) = Faux$, il ne faut pas faire confiance à m

§. $e = k$ en symétrique, e est la clef publique en asymétrique



Angles d'attaque

Objectif d'Ève : **forger** une signature.

Existential UnForgeability (EUF) : forger un couple (m, σ) est difficile

Selective UnForgeability (SUF) : à m choisi, forger un couple (m, σ) est difficile

Universal UnForgeability (UUF) : pour tout m , forger un couple (m, σ) est difficile

Key recovery resistance : trouver la clef est difficile

D'après vous ?

Lequel de ces problèmes est le plus difficile ?



Contenu Partie V : Signature

19 Introduction

20 Signature à clef privée

21 Signature à clef publique



Signature à clef privée

Dans le cas de la cryptographie symétrique (clef privée), la signature se fait avec un MAC.

Prérequis :

- 🧐 Alice et Bob doivent connaître une clef privée commune

Garanties :

- 🧐 Intégrité
- 🧐 Authenticité
- 🧐 Non-répudiation

Bah non :

non-répudiation = seul Bob peut signer

Mais Alice a la clef secrète aussi, donc Alice aussi peut signer. . .

(mais c'est pas bien grave)



Contenu Partie V : Signature

19 Introduction

20 Signature à clef privée

21 Signature à clef publique



Signature à clef publique

Les signatures à clef publique sont basées sur les mêmes problèmes difficiles que le chiffrement.

Garanties :

-  Intégrité
-  Authenticité
-  Non-répudiation



Exemple : Signature RSA

Chiffrement RSA

Clef publique : $(n = pq, e)$

Clef privée : d tq $ed = 1 \bmod \phi(n)$

Chiffrement : $m^e \bmod n$

Déchiffrement : $c^d \bmod n$

Signature RSA

Clef publique : $(n = pq, e)$

Clef privée : d tq $ed = 1 \bmod \phi(n)$

Signature : $\sigma = m^d \bmod n$

Vérification : $\sigma^e = m \bmod n$



Attaque de la signature RSA

Exercice :

Forger (EUF) la signature RSA :

🗣️ vous pouvez connaître des couples (message, signature) :

$$(m_1, \sigma_1), (m_2, \sigma_2), \dots$$

🗣️ trouvez un couple (message, signature) que vous ne connaissiez pas, i.e.

Trouvez (m, σ) avec $m \neq m_i, \forall i$

Rappel :

$$\sigma = m^d \bmod n$$



Attaque de la signature RSA

Exercice :

Forger (EUF) la signature RSA :

🗉 vous pouvez connaître des couples (message, signature) :

$$(m_1, \sigma_1), (m_2, \sigma_2), \dots$$

🗉 trouvez un couple (message, signature) que vous ne connaissiez pas, i.e.

Trouvez (m, σ) avec $m \neq m_i, \forall i$

On remarque que :

$$\sigma(m_1) \cdot \sigma(m_2) = \sigma(m_1 \cdot m_2)$$

On signe m_1 et m_2 , on connaît donc $(m_1, \sigma(m_1)), (m_2, \sigma(m_2))$.

On forge le couple $(m_1 \cdot m_2, \sigma(m_1) \cdot \sigma(m_2))$.



Attaque de la signature RSA

Exercice :

Forger (EUF) la signature RSA :

🧐 vous pouvez connaître des couples (message, signature) :

$$(m_1, \sigma_1), (m_2, \sigma_2), \dots$$

🧐 trouvez un couple (message, signature) que vous ne connaissiez pas, i.e.

Trouvez (m, σ) avec $m \neq m_i, \forall i$

On remarque que :

$$\sigma(m_1) \cdot \sigma(m_2) = \sigma(m_1 \cdot m_2)$$

On signe m_1 et m_2 , on connaît donc $(m_1, \sigma(m_1)), (m_2, \sigma(m_2))$.

On forge le couple $(m_1 \cdot m_2, \sigma(m_1) \cdot \sigma(m_2))$.

On peut éviter cette attaque en hachant m avant de signer !



Exemple : Signature ElGamal

Signature ElGamal

Paramètres : p un premier, g générateur du groupe $(\mathbb{Z}^*, \times)$, H hachage cryptographique

Clef privée : $x \in [1, p - 2]$

Clef publique : $y = g^x \bmod p$

Signature :

- 1 Choisir $k \in [2, p - 2]$ premier avec $p - 1$
- 2 Calculer $r = g^k \bmod p$
- 3 Calculer $s = (H(m) - xr)k^{-1} \bmod (p - 1)$
- 4 Signature : (r, s)

Vérification :

- 1 Vérifier que $r \in [1, p - 1]$, $s \in [1, p - 2]$
- 2 Vérifier que $g^{H(m)} = y^r r^s \bmod p$



Sixième partie VI

Certificats électroniques



Contenu Partie VI : Certificats électroniques

22 Introduction

23 Fonctionnement



Principe

Un aspect important de la cryptographie asymétrique n'a pas été traité jusqu'à maintenant :

D'après vous ?

Comment on partage la clef publique ?



Principe

Un aspect important de la cryptographie asymétrique n'a pas été traité jusqu'à maintenant :

Bob **publie sa clef publique sur un annuaire en ligne.**

Alice l'utilise pour parler à Bob. Est-ce un échange sécurisé ?



Principe

Un aspect important de la cryptographie asymétrique n'a pas été traité jusqu'à maintenant :

Bob **publie sa clef publique sur un annuaire en ligne.**

Alice l'utilise pour parler à Bob. Est-ce un échange sécurisé ?

Non ! Rien ne prouve qu'Ève n'a pas intercepté et modifié la clef publique !



Principe

Un aspect important de la cryptographie asymétrique n'a pas été traité jusqu'à maintenant :

Bob **publie sa clef publique sur un annuaire en ligne.**

Alice l'utilise pour parler à Bob. Est-ce un échange sécurisé ?

Non ! Rien ne prouve qu'Ève n'a pas intercepté et modifié la clef publique !

*La clef publique est **publique** = **non-confidentielle**, mais il faut garantir son **intégrité** !*

La solution est le **certificat électronique**.



Comment partager la clef publique ?

D'après vous ?

Comment ça marche ?



Comment partager la clef publique ?

D'après vous ?

Comment ça marche ?

- 🧐 c'est un **gros casse-tête**
- 🧐 on ne peut pas se servir de cryptographie
- 🧐 on va faire autrement, mais :
 - 🧐 ça coûte **très** cher
 - 🧐 ça implique que Alice et Bob fassent confiance à quelqu'un d'autre en plus

En pratique :

pour sécuriser une application, c'est la partie qui **prend le plus de temps**



Infrastructure de clef publique (PKI)

Notez :

- 👤 pas un algo
- 👤 pas un logiciel
- 👤 une **infrastructure**
 - 🌐 périmètre : **Internet** en entier
 - 🌐 ON A **MODIFIÉ INTERNET** POUR ÇA ! **PHYSIQUEMENT** !
 - 🌐 il y a des machines dédiées, des entreprises dont c'est le métier

Vous **devez comprendre** le fonctionnement du partage de clef publique.

Vous aurez à le faire (en stage) :

- 👤 1/3 de la promo fait un partage de clef publique à la main (shell)
- 👤 beaucoup des autres aussi, mais en interne dans un SI (pas Internet), c'est plus simple et automatisable



Définition

Certificat

Fichier électronique qui comprend :

Clef publique d'un individu/entité/domaine

Identification : nom, prénom, nom de domaine, etc

Authentification Signature par un **tiers de confiance**

Autres : usage de la clef, dates de validité, informations de révocation, ...

Tiers de confiance

Autorité de certification (CA) en charge de :

Vérifier l'identité du propriétaire de la clef publique

Créer le certificat après vérification

Signer le certificat avec la clef privée de l'autorité de certification

Liste à jour des certificats révoqués



Contenu Partie VI : Certificats électroniques

22 Introduction

23 Fonctionnement



Autorité de certification ? Quèsaco ?

- 🧐 Généralement des entreprises privées
- 🧐 Liste de standards inclus dans les navigateurs Web et systèmes d'exploitation
- 🧐 On peut ajouter les certificats qu'on veut

Une **CA** est une **entité à laquelle on choisit de faire confiance**.

Pourquoi lui faire confiance : *parce que la **valeur** de la CA, c'est la **confiance** de ses **utilisateurs**.*

⇒ si la CA trahit la confiance de ses utilisateurs, elle périlite.

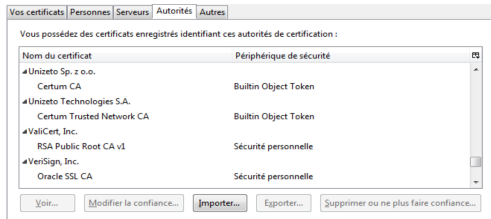
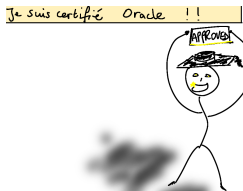


Image : magasin de certificats de Firefox



Certificat : Exemple

Exemple d'un certificat pour le site web www.france-université-numérique-mooc.fr

Les détails techniques du certificat, la clé et la signature se trouvent dans **Détails**

Général

Détails

Ce certificat a été vérifié pour les utilisations suivantes :

- Certificat client SSL
- Certificat serveur SSL

Émis pour	
Nom commun (CN)	www.france-universite-numerique-mooc.fr
Organisation (O)	<Ne fait pas partie du certificat>
Unité d'organisation (OU)	Domain Control Validated
Numéro de série	00:EE:CE:37:A0:F9:50:16:57:BC:0A:C2:4B:A8:9F:0E:41
Émis par	
Nom commun (CN)	TERENA SSL CA
Organisation (O)	TERENA
Unité d'organisation (OU)	<Ne fait pas partie du certificat>
Période de validité	
Début le	08/10/2013
Expire le	08/10/2016
Empreintes numériques	
Empreinte numérique SHA-256	6E:D0:7E:51:A4:2A:86:97:A0:A8:C0:70:9C:32:E8:8B:16:B3:89:22:A2:C5:AE:5A:FE:35:99:0E:B3:79:10:EB
Empreinte numérique SHA1	86:22:B9:4F:FB:7B:9F:45:DF:B0:89:C0:A6:C0:83:DF:F6:2E:0B:9A

Détenteur de la
clé publique

Autorité de
certification

Dates de validité du
certificat

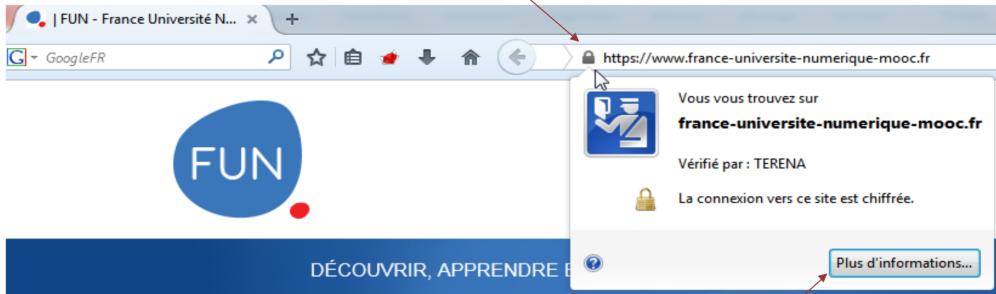


Certificat : Informations

Où trouver les certificats dans un navigateur ?

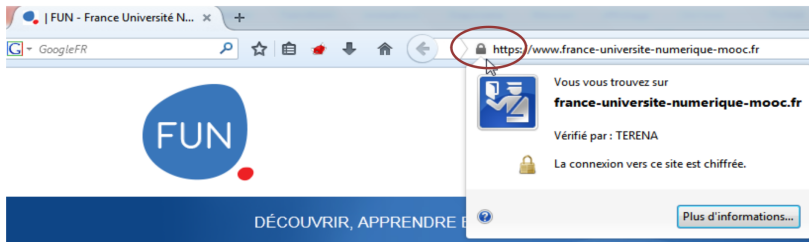
Exemple avec Firefox pour ouvrir le certificat d'un site Web.

Cliquer sur le cadenas à côté de l'URL



Cliquer ici pour afficher le certificat

Certificat : Garanties



Certificat valide $\Rightarrow$ l'utilisateur a deux avantages, caractéristiques du HTTPS :

- 🧐 confiance en la légitimité du site (certificat vérifié et signé par une CA de confiance)
- 🧐 le certificat contient la clef publique : *l'utilisateur peut chiffrer sa connexion au site*

Attention : Google et ses ambitions de contrôle d'Internet

En passant...

Google a des ambitions de contrôler le monde (pour le bourrer de pubs).

Sa nouvelle idée : **Web Environment Integrity**

<https://github.com/RubertBenWiser/Web-Environment-Integrity>

Pourquoi ? officiellement, empêcher les bots d'accéder aux sites Web (parce que c'est très très grave et que tout le monde trouve ça essentiel)

Comment ? en envoyant **tout** votre trafic Internet (oui oui, tout) à Google, et que Google décide à quels bouts de la page Web vous avez le droit d'accéder. En gros, Google voit tout d'Internet et censure ce qu'il veut.

Ils demandent à tous les sites Web de supporter ça, évidemment ne le faites pas.



Google Web Environment Integrity, comment ça marche ?

Mais techniquement comment ça marche leur horreur ?

(un bon exemple de ce qu'il ne faut pas faire)

- 🧐 chaque site Web doit avoir un canal sécurisé avec Google (certificat de Google sur tous les sites Web)
- 🧐 à chaque requête d'un utilisateur, le site Web doit envoyer (requête, détails utilisateur) à un serveur central de Google
- 🧐 Google filtre et renvoie une requête censurée que le site Web transmet



Google Web Environment Integrity, comment ça marche ?

Mais techniquement comment ça marche leur horreur ?

(un bon exemple de ce qu'il ne faut pas faire)

- 🧐 chaque site Web doit avoir un canal sécurisé avec Google (certificat de Google sur tous les sites Web)
- 🧐 à chaque requête d'un utilisateur, le site Web doit envoyer (requête, détails utilisateur) à un serveur central de Google
- 🧐 Google filtre et renvoie une requête censurée que le site Web transmet

D'après vous ?

Ça ressemble à quoi ?

Google Web Environment Integrity, comment ça marche ?

Mais techniquement comment ça marche leur horreur ?

(un bon exemple de ce qu'il ne faut pas faire)

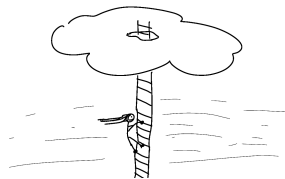
- 🧐 chaque site Web doit avoir un canal sécurisé avec Google (certificat de Google sur tous les sites Web)
 - 🧐 à chaque requête d'un utilisateur, le site Web doit envoyer (requête, détails utilisateur) à un serveur central de Google
 - 🧐 Google filtre et renvoie une requête censurée que le site Web transmet
 - 🧐 c'est une CA **en ligne**
- ⇒ ça **double** le trafic d'Internet

*Pour votre culture, l'impact écologique d'Internet n'est pas négligeable, et ce qui importe c'est le **dimensionnement**, l'usage a peu d'importance (cf. travaux d'Anne-Cécile Orgerie).*



Septième partie VII

Aller plus haut





Contenu Partie VII : Aller plus haut

24 Introduction

25 Zero-Knowledge

26 White-box

27 Multi-Party Computation

28 Fully-Homomorphic Encryption

29 Blockchain



Introduction

Dans ce qui suit : **revue** de ce qui est **faisable** avec de la cryptographie.

Retenez que ça existe !



Contenu Partie VII : Aller plus haut

24 Introduction

25 Zero-Knowledge

26 White-box

27 Multi-Party Computation

28 Fully-Homomorphic Encryption

29 Blockchain

Zero-Knowledge Proof

Zero-Knowledge Proof

Objectif : **prouver** qu'on **connaît** une information, **sans dévoiler l'information**.

Existence : majoritairement théorique, mais quelques solutions existent

Efficacité : peu efficace mais réaliste, débuts d'applications, e.g. vote électronique

Et pourtant je sais !





Contenu Partie VII : Aller plus haut

24 Introduction

25 Zero-Knowledge

26 **White-box**

27 Multi-Party Computation

28 Fully-Homomorphic Encryption

29 Blockchain



White-box

White-box cryptography

Objectif : la **cryptographie en boîte blanche** consiste à **cacher la clef secrète** contenue dans l'**implémentation d'un système cryptographique**.

Existence : pour l'instant, toutes les solutions proposées sont cassées sous quelques mois

Remarque : Nécessaire en pratique. Les solutions *cryptographiques* échouent → pas de preuve formelle.

Pour le moment, on utilise de l'**offuscation** à la place : en gros, on bidouille.

Exemples : compilation compliquée du programme, retirer les commentaires, retirer les headers de fichiers, cross-compiler entre langages (plusieurs étapes avant un exécutable), utiliser un packer (souvent UPX), compresser sans perte, etc

Q : Quel principe fondamental est remis en cause dans le contexte de la boîte blanche ?



Contenu Partie VII : Aller plus haut

24 Introduction

25 Zero-Knowledge

26 White-box

27 **Multi-Party Computation**

28 Fully-Homomorphic Encryption

29 Blockchain



Multi-Party Computation

Multi-Party Computation

Objectif : le **calcul multipartite** sert à n parties $p_1, \dots, p_n$ de données respectives $d_1, \dots, d_n$, à calculer une même fonction $F(d_1, \dots, d_n)$ sans que les autres parties n'obtiennent d'autre information sur leur donnée.

Existence : des solutions existent

Efficacité : peu efficace mais réaliste, quelques utilisations industrielles

Une clef pour toi, une pour moi,
une pour tata...





Contenu Partie VII : Aller plus haut

24 Introduction

25 Zero-Knowledge

26 White-box

27 Multi-Party Computation

28 Fully-Homomorphic Encryption

29 Blockchain



(Fully) Homomorphic Encryption

(Fully) Homomorphic Encryption

Objectif : le **chiffrement (complètement) homomorphe** sert à **déléguer un calcul (coûteux)** à une entité (le **Cloud** par exemple) **sans révéler d'information sur les données manipulées**.

Existence : des solutions existent

Efficacité : réaliste, débuts d'applications industrielles. Utilisé pour le vote électronique.



Contenu Partie VII : Aller plus haut

24 Introduction

25 Zero-Knowledge

26 White-box

27 Multi-Party Computation

28 Fully-Homomorphic Encryption

29 Blockchain



Blockchain

Blockchain

Objectif : le **chaînage de blocs** est une solution de **gestion de données décentralisée**.

But : intégrité des données sans autorité centrale : chaque participant a l'historique des modifications de toutes les données (la valeur des données est fixée par la **majorité**).

Existence : des solutions existent

Efficacité : coût monumental, mais beaucoup utilisé (souvent dans des cas où des solutions plus efficaces existent) ^a

Sécurité : défaut fondamental ; la majorité, ça s'obtient ! (voir après)

^a. Question de buzz. . .



Blockchain - cryptomonnaies

Cryptomonnaie

Objectif : Monnaie décentralisée (pas plus de cryptographie que dans d'autres monnaies).

Pourquoi : ne pas faire confiance à une autorité centrale (banque centrale).



Blockchain - cryptomonnaies

Cryptomonnaie

Objectif : Monnaie décentralisée (pas plus de cryptographie que dans d'autres monnaies).

Pourquoi : ne pas faire confiance à une autorité centrale (banque centrale).

Et c'est bien ?



Blockchain - cryptomonnaies

Cryptomonnaie

Objectif : Monnaie décentralisée (pas plus de cryptographie que dans d'autres monnaies).

Pourquoi : ne pas faire confiance à une autorité centrale (banque centrale).

Bof. . .- autorité = majorité.

Banque centrale : peut être soumise à la loi, contrôles.

Majorité : incontrôlable.

→ discutable



Blockchain - sécurité

Sécurité d'une blockchain

Principe : la majorité décide de l'état actuel des données (e.g. comptes de monnaies).

Un attaquant : casser l'intégrité de la majorité (+++ collisions sur du hachage)

→ impossible

Mais... pourquoi un seul attaquant ?

Défaut fondamental ; la majorité, ça s'obtient !

e.g. : identité numérique (accès informatique par ID, contrôlé par un gouvernement).

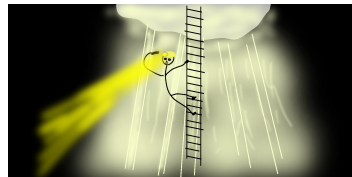
- 🗣️ Chine : avant de bannir bitcoin ^a le gouvernement avait la majorité.
- 🗣️ Inde : le gouvernement contrôle 1 milliard+ de comptes numériques.
- 🗣️ R-U : en train d'instaurer l'identité numérique.
- 🗣️ UE : des mesures similaires sont proposées (Chat Control pour l'instant).

^a. coûtant trop de centrales nucléaires



Huitième partie VIII

Aller plus bas





Contenu Partie VIII : Aller plus bas

30 Introduction

31 Générateurs pseudo-aléatoires

32 Oblivious RAM

33 Attaques par canaux auxiliaires

34 HSM

Introduction

Petite plongée au niveau matériel

D'après vous ?

Quelles sont les problématiques au niveau matériel ?

D'après vous ?

Quelles sont les solutions de sécurité informatique matérielles ?



Contenu Partie VIII : Aller plus bas

30 Introduction

31 Générateurs pseudo-aléatoires

32 Oblivious RAM

33 Attaques par canaux auxiliaires

34 HSM

Générateurs aléatoires

L'intégralité de la cryptographie (et donc de la SSI) repose sur l'utilisation de **grandes quantités d'aléa** !

Aléa jacta est ! *



L'aléa est à
gérer. *

Générateurs aléatoires

L'intégralité de la cryptographie (et donc de la SSI) repose sur l'utilisation de **grandes quantités d'aléa** !

Mais **comment génère-t-on de l'aléa** ?

D'après vous ?

Vous feriez comment ?

Aléa jacta est ! *



L'aléa est à jeter.*

Générateurs aléatoires

L'intégralité de la cryptographie (et donc de la SSI) repose sur l'utilisation de **grandes quantités d'aléa** !

Mais **comment génère-t-on de l'aléa** ?

Aléatoire 'parfait'

L'aléatoire dit '**parfait**' ne **peut pas être distingué d'une distribution uniforme** avec les outils actuels.

Exemples : dernières décimales de l'heure, de la température du CPU, ...

Efficacité : la génération d'aléa 'parfait' est **très lente**.

Aléa jacta est ! *



L'aléa est à jeter.*

Générateurs pseudo-aléatoires

Pour générer **assez d'aléa efficacement**, on utilise des **générateurs pseudo-aléatoires**.

Générateur pseudo-aléatoire

Objectif : à partir de n bits d'aléa parfait, générer $m > n$ bits de pseudo-aléa.

Exemples :

chiffrement symétrique en particulier les chiffrements à flot ou 4 tours d'AES

générateurs dédiés plus efficaces

Efficacité : la génération de pseudo-aléa est **lente**.



Contenu Partie VIII : Aller plus bas

30 Introduction

31 Générateurs pseudo-aléatoires

32 Oblivious RAM

33 Attaques par canaux auxiliaires

34 HSM

Oblivious RAM

Problématique : un programme (e.g. sur un serveur) **sait à quelles adresses mémoire il accède**

→ il peut savoir où se trouvent les données sensibles.

Oblivious RAM

Objectif : cacher quel accès mémoire est sensible.

Principe : une **Oblivious Random Access Machine** est un compilateur qui transforme un logiciel qui ne cache pas les accès sensibles en logiciel qui cache les accès sensibles.

Existence : quelques solutions existent

Efficacité : coûteux

Je vous jure, je sais plus !!





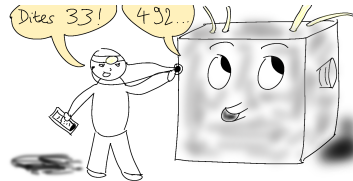
Contenu Partie VIII : Aller plus bas

- 30 Introduction
- 31 Générateurs pseudo-aléatoires
- 32 Oblivious RAM
- 33 Attaques par canaux auxiliaires**
- 34 HSM

Principe

Modèles d'attaquante :

Théorique (boîte noire) : Ève a accès aux entrées/sorties du cryptosystème



Principe

Modèles d'attaquante :

Théorique (boîte noire) : Ève a accès aux entrées/sorties du cryptosystème

Réal (boîte grise) : Ève a accès à une implémentation du cryptosystème

→ elle peut obtenir :



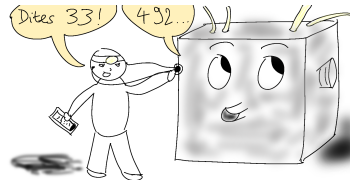
entrées/sorties



mesures physiques : temps de calcul, consommation d'énergie, émissions électro-magnétiques, ...



autres : corruption mémoire, optimisations d'OS.
instructions exécutées par le processeur, ...



Principe

Modèles d'attaquante :

Théorique (boîte noire) : Ève a accès aux entrées/sorties du cryptosystème

Réal (boîte grise) : Ève a accès à une implémentation du cryptosystème

→ elle peut obtenir :



entrées/sorties

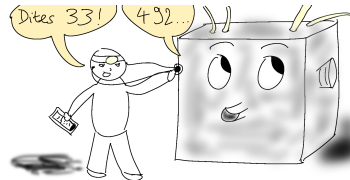


mesures physiques : temps de calcul, consommation d'énergie, émissions électro-magnétiques, ...



autres : corruption mémoire, optimisations d'OS.
instructions exécutées par le processeur, ...

Problématique : ces mesures physiques dépendent
des valeurs (sensibles) manipulées → information supplémentaire



C'est grave docteur ?

C'est grave docteur ?



C'est grave docteur ?

C'est grave docteur ?

Tous les cryptosystèmes non-protégés contre les attaques physiques sont vulnérables à une **récupération de clef en une fraction de seconde**.

Note : Ève doit avoir un **accès physique** et pouvoir faire efficacement des **mesures physiques du cryptosystème**

→ pas toujours applicable

→ très applicable sur les objets connectés !

Screaming channels : des attaques similaires sur les objets connectés via la bluetooth et le Wifi (peu étudié)

Protection

Objectif : rendre les mesures indépendantes des données sensibles :

temps de calcul → rendre le temps indépendant des données sensibles (**pire temps**)

énergie, ondes, ... ajouter beaucoup de **redondance** et d'**aléa**

⇒ énorme amplification de coût : $\times 1000$ en logiciel, $\times 10$ en matériel !

En pratique :

- 🧐 on protège **un peu** les implémentations logicielles sur grosses machines (on espère que les mesures sur les données sensibles sont noyées dans les autres données)
- 🧐 on **peut** protéger **bien** les implémentations matérielles sur objets connectés **mais** la protection est **rare**
- 🧐 on est **incapable** de **protéger des implémentations logicielles sur objets connectés (micro-contrôleurs)** 🚩

C'est gênant car c'est la très grande majorité.

🚩. Pour l'instant.



Contenu Partie VIII : Aller plus bas

30 Introduction

31 Générateurs pseudo-aléatoires

32 Oblivious RAM

33 Attaques par canaux auxiliaires

34 HSM

Hardware Security Module

Hardware Security Module (HSM)

Principe : une **boîte noire transactionnelle** est un matériel électronique contenant du matériel cryptographique (clefs, cryptosystèmes, ...). Généralement, c'est une carte électronique PCI ou un micro-contrôleur dédié.

Garanties :

- 🔒 toute la cryptographie se fait localement dans le module
- 🔒 le module a une armure (physique) pour l'isoler
- 🔒 le module a des mécanismes de détection d'intrusion
- 🔒 le module peut détruire les données sensibles en cas d'intrusion
- 🔒 authentification de l'OS et des logiciels avant exécution

Pas impénétrable, mais difficile à pénétrer.



Neuvième partie IX

Aller plus loin



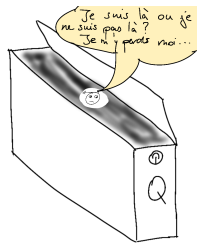
Contenu Partie IX : Aller plus loin

35 Cryptographie post-quantique



Principe

L'ordinateur quantique arrive !



||. Lentement : ça fait 40 ans qu'on nous dit "c'est dans 20 ans".

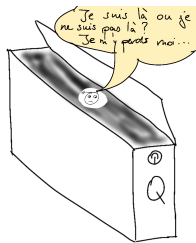
Principe

L'ordinateur quantique arrive !

Conséquence : l'attaquante a un outil supplémentaire.

Il faut :

1. évaluer la puissance de cet outil
2. prévoir des parades



||. Lentement : ça fait 40 ans qu'on nous dit "c'est dans 20 ans".



Question

Q : Vous avez déjà entendu parler de cryptographie quantique ? Post-quantique ? Qu'en savez-vous ?



Recherche exhaustive – Sous-groupe caché

Prévu : l'ordinateur quantique améliorerait la résolution de **2** problèmes :

Recherche exhaustive

L'**algorithme de Grover** donne une solution probabiliste à la **recherche exhaustive** en $\mathcal{O}(\sqrt{N})$ opérations plutôt que N pour un ensemble de taille N .

L'accélération est donc **quadratique**.

Sous-groupe caché (*Abélien*)

L'**algorithme de Schor** résout la **factorisation d'un entier de n bits** en temps polynomial $\mathcal{O}((n^2)(\log(n))(\log(\log(n))))$ (le meilleur algo. classique est sous-exponentiel).

L'**algorithme de Simon** trouve la **période** d'une fonction périodique en $\mathcal{O}(n)$ opérations plutôt que $\Omega(2^{n/2})$ pour une fonction de n vers n bits.

Factorisation et *période* sont des sous-cas du problème *trouver le sous-groupe caché* d'un groupe **Abélien**. Ils donnent une accélération **exponentielle**.

Non-Abélien casse **les réseaux (lattices)**. L'accélération est pour l'instant **quadratique**.



Conséquences sur les problèmes cryptographiques

Recherche exhaustive

La recherche exhaustive accélérée quadratiquement change **tout** à l'algorithmique en général. En cryptographie, la première conséquence est qu'il faut **doubler la taille des clefs**.

Sous-groupe caché

Abélien : résout en temps **polynomial**

- 🕒 la factorisation des nombres entiers
- 🕒 le logarithme discret
- 🕒 le mode Even-Mansour (xor – Encrypt – xor) et beaucoup d'autres modes (chiffrement symétrique et MAC)

Non-Abélien : accélérée quadratiquement (et peut-être plus) résout

- 🕒 l'isomorphisme de graphes
- 🕒 les problèmes de réseaux (lattices)

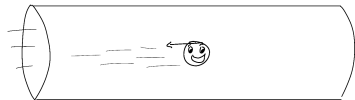


Sécurité côté symétrique

- 🧐 il faut doubler la taille des clefs
- 🧐 il faut choisir/créer des modes opératoires qui le prennent en compte
- 🧐 autres ? c'est en étude

Solution actuelle : Saturnin-CTR-Cascade ou Saturnin-QCB.

Spinning like a qubit!!





Sécurité côté asymétrique

- 👤 on abandonne certains axes
 - 👤 focalisation sur les problèmes de
 - 🔍 réseaux (lattices)
 - 🔍 codes correcteurs
 - 🔍 systèmes multivariés
- a priori post-quantiques



Dixième partie X

Résumé global sur la cryptographie



Contenu Partie X : Résumé global sur la cryptographie

36 La cryptographie utilisée en vrai sur Internet

37 Résumé général



Canal hybride : l'exemple de TLS

En communication, la **cryptographie** sert à construire (et maintenir) un **canal sécurisé**.

Exemples :

- 🗣️ canal sécurisé téléphonique (3G, 4G, 5G)
- 🗣️ canal sécurisé Web (https = http avec TLS)
- 🗣️ canal sécurisé mail (smtp avec TLS ou autre)
- 🗣️ canal sécurisé shell (SSH)
- 🗣️ canal sécurisé d'accès distant (RDP ou SSH)
- 🗣️ canal sécurisé global = VPN (via TLS ou autre)
- 🗣️ ...

Le plus souvent c'est un **canal hybride** ^{**}.

On va détailler les étapes dans TLS.

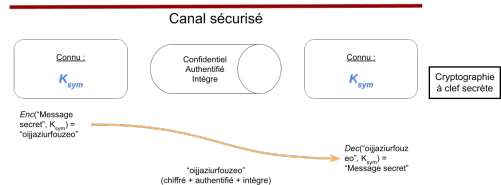
******. Cas exceptionnel : quand on a pu partager une clef symétrique à l'avance (e.g. une entreprise qui déploie un capteur loin de ses bureaux). Alors on peut directement faire du symétrique.

Canal sécurisé symétrique

Secret commun K_{sym}

⇒ canal sécurisé symétrique

- 🔒 symétrique
⇒ pas cher
- 🔒 sécurisé = Confidentiel,
Authentifié et Intègre

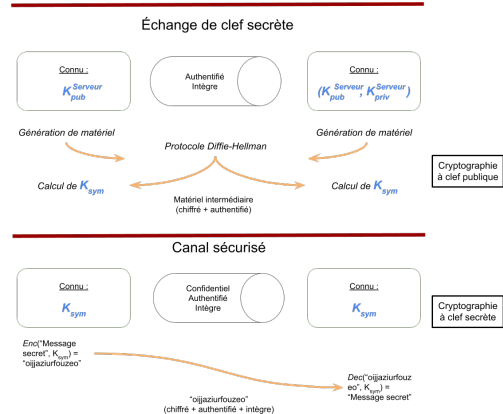


Échange de clef

Échanger K_{sym} ?
Possible avec $K_{pub}^{Serveur}$ en commun

🧠 asymétrique
⇒ un peu cher

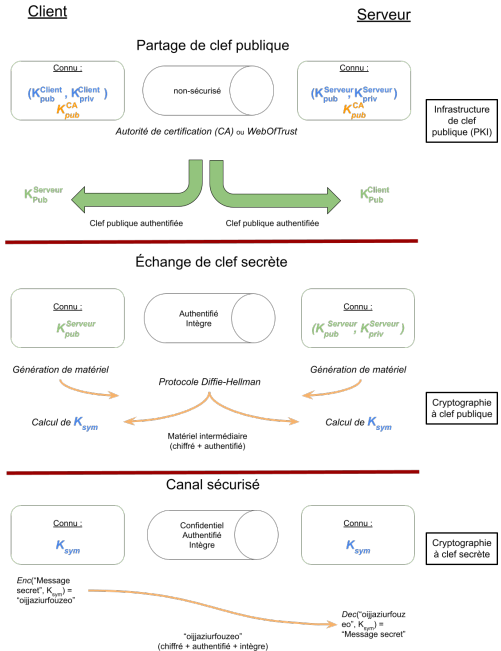
🧠 $K_{pub}^{Serveur}$ permet d'authentifier
i.e. éviter les Man-in-the-Middle



Infrastructure de clef publique

Partager $K_{pub}^{Serveur}$ avec
intégrité + authentification ?
Par un tiers de confiance (CA).

- repose sur la confiance
- demande une infrastructure dédiée



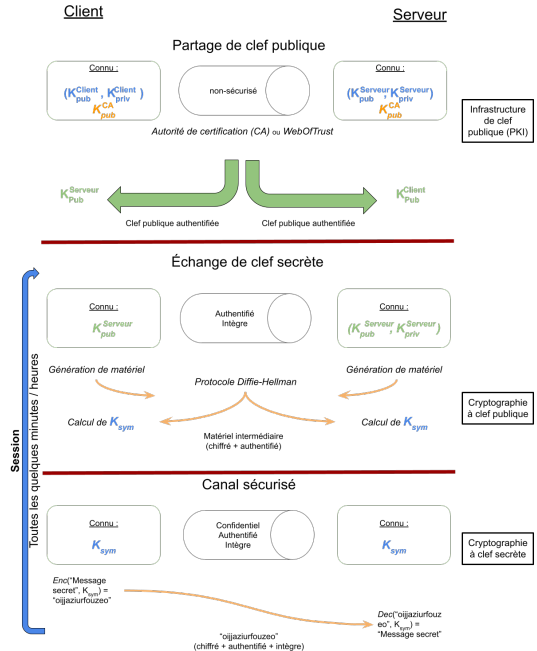
Session

Et si...

- 🧐 K_{sym} est compromise ?
- 🧐 on échange trop de données avec la même clef ?

⇒ changer de clef régulièrement
(secondes/minutes/heures)

- 🧐 session courte
- 🧐 K_{sym} différente à chaque session



Détails de la PKI

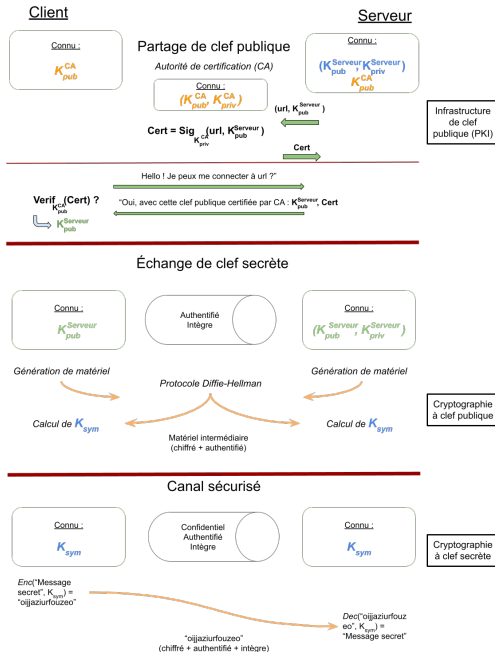
Crée un canal authentifié et intègre.

Contraintes :

- Client ne connaît que K_{pub}^{CA}
- CA hors-ligne

Protocole :

- hors-ligne : Serveur obtient Certificat = $\text{Sig}_{K_{priv}^{CA}}(\text{url}, K_{pub}^{Serveur})$
- en ligne : serveur donne $K_{pub}^{Serveur}$ et le certificat
- Client vérifie la signature avec K_{pub}^{CA}
- (option : Client partage K_{pub}^{Client})



Diffie-Hellman

Ajoute la confidentialité à un canal authentifié et intègre.

Principe :

avoir $K_{sym} = g^{ab} \mod p$

K_{sym} ne circule jamais

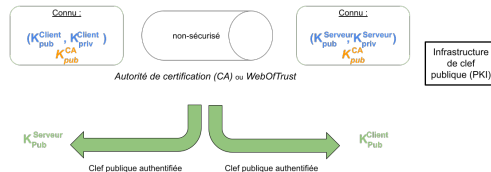
sécurité : logarithme discret difficile
 $(g^a \mod p \mapsto a)$

$K_{pub}^{Serveur} \Rightarrow$ Authentification + Intégrité

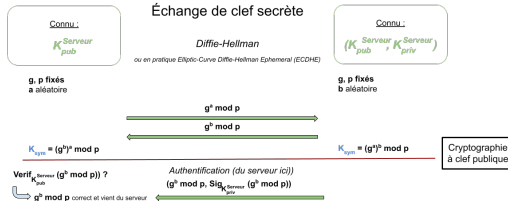
Client

Serveur

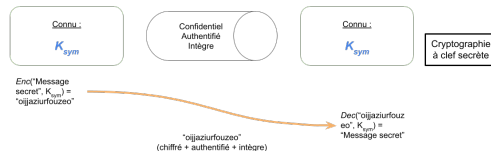
Partage de clef publique



Échange de clef secrète



Canal sécurisé



Chiffrement authentifié

=

Canal sécurisé

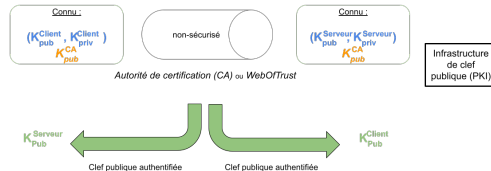
Permet d'envoyer beaucoup de messages, efficacement, sur un canal authentifié, intègre et confidentiel.

- 🧠 chiffré $C \Rightarrow$ Confidentialité
- 🧠 tag $T \Rightarrow$ Authentification, Intégrité
- 🧠 envoyer $\sim 2^{40}$ bits de message avec la même clef

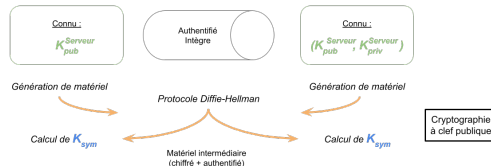
Client

Serveur

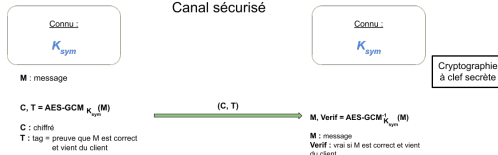
Partage de clef publique



Échange de clef secrète



Canal sécurisé

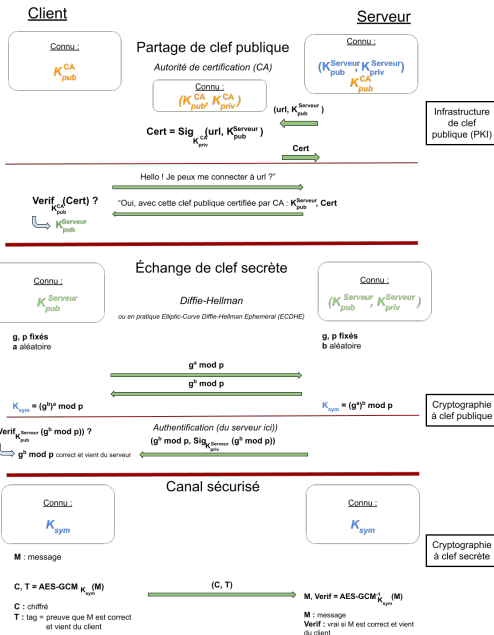


TLS détaillé - simplifié

- création de $(K_{priv}^{Serveur}, K_{pub}^{Serveur})$ et obtention du certificat par le dev Web
- le reste est automatique

cf. TLS v1.3 complet :

<https://www.rfc-editor.org/rfc/rfc8446>



Outils cryptographiques utilisés

Outils :

- 🧑 Signatures par la CA (asymétrique)
- 🧑 Signatures par le serveur (et éventuellement le client) (asymétrique)
- 🧑 Échange de clef (asymétrique)
- 🧑 Chiffrement authentifié (symétrique)

Standards en pratique (TLS 1.3) :

- 🧑 au choix de la CA : souvent signature RSA avec SHA2-256, DSA ou ECDSA
- 🧑 ECDSA (Elliptic-Curve Digital Signature Algorithm)
- 🧑 ECDHE (Elliptic-Curve Diffie-Hellman Ephemeral)
- 🧑 AES[128/192/256] en mode GCM ††

††. ou Chacha20-Poly1305 rarement



Statistiques d'une connexion HTTPS

Support	TLS	Key-exchange	Chiffrement	Authentification	Hachage
92.94%	v1.2	ECDHE + RSA-sig	AES128-GCM	AES128-GCM	SHA256
92.90%	v1.2	ECDHE + RSA-sig	AES256-GCM	AES128-GCM	SHA384
74.83%	v1.2	ECDHE + RSA-sig	ChaCha20-Poly1305	ChaCha20-Poly1305	SHA256
73.18%	v1.3	ECDHE + MAC	AES256-GCM	AES256-GCM	SHA384
...					
66.23%	v1.2	ECDHE + RSA-sig	AES256-CBC	AES256-CBCMAC	SHA384
31.22%	v1.1	ECDHE + RSA-sig	AES256-CBC	AES256-CBCMAC	SHA
27.32%	v1.0	ECDHE + RSA-sig	AES256-CBC	AES256-CBCMAC	SHA

Key-exchange : ECDHE + ECDSA (27%)

Modes : CCM (5%)

Chiffrements : 3DES (19%), Aria, Camellia (4%), SEED, IDEA, RC4 (1%).

ECDHE : Elliptic-Curve Diffie-Hellman Ephemeral. Seul ECDHE est supporté dans TLS v1.3.



Statistiques d'une connexion HTTPS

Support	TLS	Key-exchange	Chiffrement	Authentification	Hachage
92.94%	v1.2	ECDHE + RSA-sig	AES128-GCM	AES128-GCM	SHA256
92.90%	v1.2	ECDHE + RSA-sig	AES256-GCM	AES128-GCM	SHA384
74.83%	v1.2	ECDHE + RSA-sig	ChaCha20-Poly1305	ChaCha20-Poly1305	SHA256
73.18%	v1.3	ECDHE + MAC	AES256-GCM	AES256-GCM	SHA384
			...		
66.23%	v1.2	ECDHE + RSA-sig	AES256-CBC	AES256-CBCMAC	SHA384



Statistiques d'une connexion HTTPS

Support	TLS	Key-exchange	Chiffrement	Authentification	Hachage
92.94%	v1.2	ECDHE + RSA-sig	AES128-GCM	AES128-GCM	SHA256
92.90%	v1.2	ECDHE + RSA-sig	AES256-GCM	AES128-GCM	SHA384
74.83%	v1.2	ECDHE + RSA-sig	ChaCha20-Poly1305	ChaCha20-Poly1305	SHA256
73.18%	v1.3	ECDHE + MAC	AES256-GCM	AES256-GCM	SHA384
...					
66.23%	v1.2	ECDHE + RSA-sig	AES256-CBC	AES256-CBCMAC	SHA384

Statistiques de l'étude de Matthieu Amet sur 7323 sites Web.

Note : statistiques de support, pas de choix le plus utilisé (mais c'est le 1er de la liste).



Contenu Partie X : Résumé global sur la cryptographie

36 La cryptographie utilisée en vrai sur Internet

37 Résumé général

D'après vous ?

On fait les comptes :

- 💡 Combien de domaines avons-nous déjà parcouru au sujet de la sécurité informatique ? (il y a l'informatique bien sûr, mais quoi d'autre ?)
- 💡 Est-il raisonnable d'imaginer qu'une seule ou quelques personnes sont en mesure de maîtriser tous ces domaines ?

Si même la question est
DIFFICILE Gfzf!





Résumé - 1

Cryptographie

La **cryptographie** est une boîte à outils de fonctions mathématiques (algorithmes) qui, bien utilisés, permettent d'obtenir un très haut niveau en **C**onfidentialité, **I**ntégrité, **P**reuve et **A**uthentication.



Résumé - 1

Cryptographie

La **cryptographie** est une boîte à outils de fonctions mathématiques (algorithmes) qui, bien utilisés, permettent d'obtenir un très haut niveau en **C**onfidentialité, **I**ntégrité, **P**reuve et Authentification.

Difficile : ne pas réinventer des cryptosystèmes soi-même (ou alors faire une thèse).



Résumé - 1

Cryptographie

La **cryptographie** est une boîte à outils de fonctions mathématiques (algorithmes) qui, bien utilisés, permettent d'obtenir un très haut niveau en **C**onfidentialité, **I**ntégrité, **P**reuve et **A**uthentification.

Difficile : ne pas réinventer des cryptosystèmes soi-même (ou alors faire une thèse).

Sécurité des cryptosystèmes

- 🧐 la sécurité inconditionnelle est possible, mais presque toujours inutilisable,
- 🧐 la cryptographie moderne repose sur la sécurité calculatoire : il faut des ressources délirantes pour casser un cryptosystème bien conçu.
- 🧐 les cryptosystèmes sont analysés : on a des preuves sur des morceaux, pas sur l'ensemble. On teste un système pendant des années avant de l'utiliser.



Résumé - 2

Types de cryptosystèmes

- 🧠 symétrique (*a.k.a.* clef secrète) : efficace et pas cher, permet de prolonger indéfiniment ^a un canal sécurisé *existant*,
- 🧠 asymétrique (*a.k.a.* clef publique) : cher, permet de la sécurité exotique et de créer un canal sécurisé à partir d'un canal non-sécurisé,
- 🧠 hybride : on combine les deux

^a. pour avoir de la forward-secrecy, on a besoin de sessions et de clef publique.



Résumé - 2

Types de cryptosystèmes

- 🧠 symétrique (*a.k.a.* clef secrète) : efficace et pas cher, permet de prolonger indéfiniment ^a un canal sécurisé *existant*,
- 🧠 asymétrique (*a.k.a.* clef publique) : cher, permet de la sécurité exotique et de créer un canal sécurisé à partir d'un canal non-sécurisé,
- 🧠 hybride : on combine les deux

^a. pour avoir de la forward-secrecy, on a besoin de sessions et de clef publique.

La taille des clefs est importante pour la sécurité (e.g. attaque par force brute), mais n'est pas le seul paramètre important (e.g. Sweet32, basé sur la taille de bloc).



Résumé - 3

Cryptosystèmes et critères de sécurité

	Chiffrement	MAC/Signature	Hachage
Confidentialité	✓		
Intégrité		✓	✓
Authentification		✓	
Preuve	✓	✓	✓



Résumé - 3

Cryptosystèmes et critères de sécurité

	Chiffrement	MAC/Signature	Hachage
Confidentialité	✓		
Intégrité		✓	✓
Authentification		✓	
Preuve	✓	✓	✓

- 🧐 le hachage n'utilise pas de clef
- 🧐 MAC/Signature : souvent Chiffrement + Hachage
- 🧐 **Création de canal sécurisé** : avec un algorithme (à clef publique) d'échange de clefs (secrètes), en pratique ECDHE.



Résumé - 4

Cryptosystèmes en pratique

 dans tous les systèmes informatiques,

a. Le partage de clef publique est abordé au Ch. 7 : PKI.



Résumé - 4

Cryptosystèmes en pratique

- 💡 dans tous les systèmes informatiques,
- 💡 protocole sécurisé majeur : **TLS 1.3**,

a. Le partage de clef publique est abordé au Ch. 7 : PKI.



Résumé - 4

Cryptosystèmes en pratique

- 🧐 dans tous les systèmes informatiques,
- 🧐 protocole sécurisé majeur : **TLS 1.3**,
- 🧐 rarement besoin d'y toucher : il faut juste **initialiser** les outils cryptographiques pour TLS : il faut **générer une paire** (clef publique, clef privée), et **partager la clef publique**^a.

^a. Le partage de clef publique est abordé au Ch. 7 : PKI.



Références liées au chapitre

<https://cyber.gouv.fr/publications/mecanismes-cryptographiques>