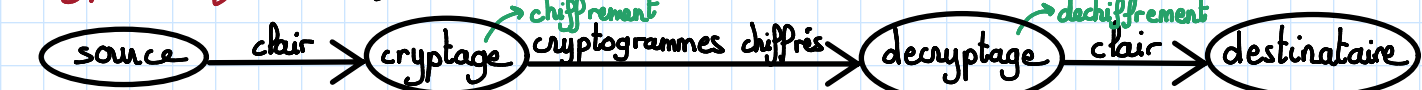


# I INTRODUCTION

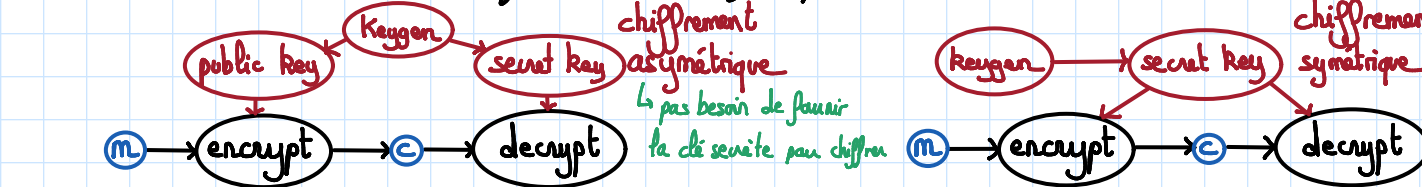
cryptanalyse (analyse de message) ≠ cryptologie (science du secret)



- propriétés de sécurité:
- confidentialité: impossible de déduire de l'info à partir du chiffré sur le clair
  - intégrité: on peut détecter que le chiffré a été modifié (hachage par exemple)
  - authentification: toute usurpation peut être détectée
  - non-répudiation: on ne peut pas nier envoyer un message qu'on a émis.

- types de canaux:
- non sûr: écoute et modification du message possible
  - sûr: écoute et modification du message impossible
  - confidentiel: écoute du message impossible
  - authentique: modification du message impossible

NIST/IETF barge  
ISO → pdr c'est nage



propriétés des chiffrements:  $C \leftarrow \text{Enc}(S_K, m)$ ,  $m \leftarrow \text{Dec}(S_K, C)$ ,  $V_{m,s} \leftarrow \text{Dec}(S_K, \text{Enc}(S_K, m)) = m$ , sécurités

chiffrement parfaitement sûr:  $H(c|m) = H(c)$ ,  $|K| \geq |M| \Rightarrow$  l'attaque de l'espace de clés est au moins aussi grand que l'attaque de l'espace de message.

clé de 128b

one time pad: clair  $\oplus$  clé  $\oplus$  clé = clair, pour un chiffré: tout clair est possible!!! parfaitement sûr mais

chiffrement de Vernam: OTP  $\Rightarrow$  malaisabilité: faire un XOR sur le chiffré ~ faire un XOR sur le clair une fois décrypté

clé non réutilisable:  $C_1 \oplus C_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2$

## II CHIFFREMENT SYMETRIQUES

- principalement substitution et transposition
- substitution: chiffrement par décalage (ex: César), substitution mono-alphabétique → attaques fréquentielles
  - substitution poly-alphabétique: Vigenère (clé + tableau 2D) → Vernam en est un cas particulier
  - substitution polygraphique: chiffrement de Playfair → on chiffre "GC" par "AI", si on veut chiffrer "AA" on insère un "X", on chiffre "AXA"
- chiffre de Hill
- $$\begin{pmatrix} C_K \\ C_{K+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} P_K \\ P_{K+1} \end{pmatrix} \pmod{26}$$
 avec  $P_K$  le  $K^{\text{ème}}$  caractère du message,  $C_K$  le  $K^{\text{ème}}$  caractère du chiffré
- nombre de solutions  $157\ 248 \approx 2^{13,22} \Rightarrow \lambda = 13,26$
- utilisé dans AES btw
- niveau de sécurité faible

- Permutation/transposition: changement d'ordre des symboles → permutation à croix grecques, permutation à tableau complet

Auguste Kerckhoffs → la cryptosystème doit être inviolable en pratique même s'il ne l'est pas dans les faits

aucun secrets dans le cryptosystème (connaître le cryptosystème ne doit pas compromettre)

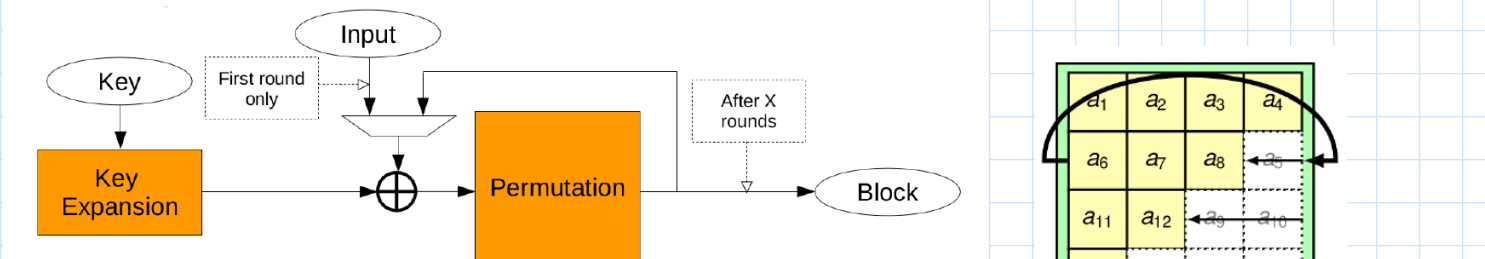
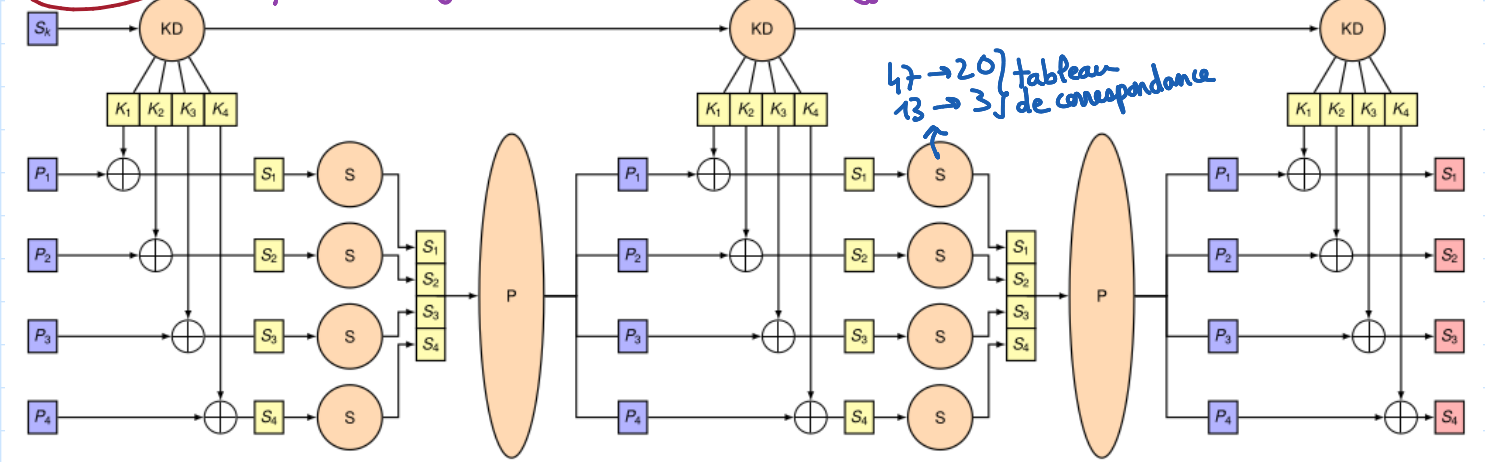
Machine de Lucifer (IBM/1960) → renforcer la sécurité grâce à une opération non-linéaire (S-BOX par ex)

confusion: propriété consistant à rendre la relation entre la clé et le chiffré la plus compliquée possible

diffusion: propriété où la redondance statistique dans un clair est dissipée dans le texte statistique du chiffré

effet d'avalanche: dans un chiffrement avec bonne diffusion, l'inversion d'un seul bit en entrée doit statistiquement pouvoir changer n bits en sortie (critère d'avalanche strict: 50% des bits)

AES post quantum-proof avec une clé de taille suffisante (256 bits min.)



- ↳ S-BOX: non linéaire, substitue, permet de faire de la confusion
- ↳ P-BOX: linéaire, mixe, permet de faire de la diffusion (= shift Rows + MixColumns)
- ↳ KD: key derivation algorithm (taille de la clé = taille du bloc)
- ↳ S+P+KD = 1 tour dans le réseau SP

|         | Pourquoi de clé | niveau de sécurité | temps |
|---------|-----------------|--------------------|-------|
| AES-128 | 128             | ≈ 126              | 10    |
| AES-192 | 192             | ≈ 190              | 12    |
| AES-256 | 256             | ≈ 254              | 14    |

quelques fois la taille de la clé, en AES on chiffre 128 bits de messages!!!

pour chiffrer des messages de grande taille: Electronic Code Book, Cipher Block Chaining, Galois Counter Mode

- ↳ ECB (naïf) → si on a le même bloc en entrée on aura la même sortie (déterministe)
- ↳ CBC (moyen) → on chiffre un bloc avec une clé et le chiffré du bloc précédent (on chiffre  $m_i \oplus c_{i-1}$ )
- ↳ GCM (pas vulnérable) → chiffrement de bloc pour générer un flux de clés + chiffrement de Vernam avec ce flux de clés

How to evaluate security?

- ↳ IND-KPA: known plain-text attack →  $\{m_i, \text{Enc}(m_i)\}$  known
- ↳ IND-CPA: chosen plain-text attack → choose  $m_i$  and asks to encrypt  $m_i$
- ↳ IND-CCA: chosen ciphertext attack → can also select  $c_i$  and ask to decrypt  $c_i$  (+ CPA)

ECB/IND-CPA?

$m_i = [\text{hello}][\text{world}] \Rightarrow \text{Enc}(m_i) = [c_0][c_1]$  on chiffre le  $m_i$  bloc  $\Rightarrow$  ECB claqué  $A_{\text{CPA}} = \frac{1}{2}$  (l'adversaire en CPA gagne toujours)

$m_j = [\text{hello}][\text{world}] \Rightarrow \text{Enc}(m_j) = [c_0][c_1]$  avec la même clé

globalement les algo déterministes sont sensibles au rejet

Combien de bloc on peut chiffrer avant de compromettre la clé: CBC theorem

VL > 0 (longueur du message en bloc), si la PRP E est sémantiquement sûre sur  $(K, X)$ , alors E utilisé en mode CBC ( $E_{\text{CBC}}$ ) est sémantiquement sûr sous CPA sur  $(K, X^L, X^{L+1})$ . Pour un adversaire effectuant q requêtes:

$$A(E_{\text{CBC}}) \leq 2A(E) + \frac{q^2 L^2}{|X|}$$

cas d'AES:  $|X| = 2^{128}$  taille de l'output AES, avantage de l'adversaire:  $2^{-80} = q^2 L^2 \Rightarrow qL = 2^{-80+128}$

$qL = 2^{48}$  blocs chiffrés, un bloc fait 128b = 16  $\times 8 = 2^4$  donc on ne doit pas aller au-delà de  $2^{23}$  octets sinon la probabilité de trouver une collision (donc de casser le chiffrement) devient supérieure à l'objectif de sécurité

# III FONCTIONS DE HACHAGES ET INTEGRITÉ

propriétés souhaitées du Hash:

- shortest check code
- robustness against bit flip → effet d'avalanche
- avoid same hash from 2 différents input: collisions

crypto classique 1/2



condensat = Hash/empreinte

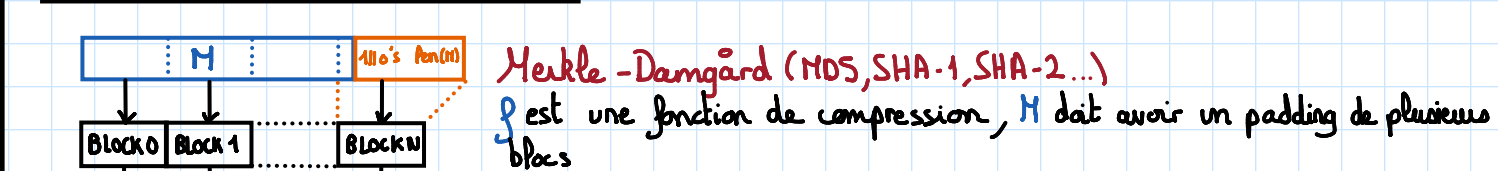
- principe:
- trouver  $m_2$  tel que  $H(m_2) = H(m_1)$  est difficile connaissant  $H(m_1)$  → first pre-image resistance
  - trouver  $m_2$  tel que  $H(m_1) = H(m_2)$  est difficile pour un  $m_1$  donné → weak collision resistance
  - trouver  $m_1$  et  $m_2$  tel que  $H(m_1) = H(m_2)$  est difficile → strong collision resistance (collision resistance)

First pre-image attack → second pre-image attack (weak collision...) → collision attack → l'autre sens est faux en général

Paradoxe des anniversaires → proba d'1/30 née un jour donné:  $1 - \left(\frac{365}{365}\right)^{30} = 7,9\%$ , proba de 2 pareil minimum:  $1 - \frac{365!}{(365-30)! 365^{30}} = 70\%$

| size of Hash | H                    | n(p;H) = 50%       |
|--------------|----------------------|--------------------|
| 16 bits      | 65 536               | 300                |
| 64 bits      | $1,8 \cdot 10^{19}$  | $3,1 \cdot 10^5$   |
| 512 bits     | $1,3 \cdot 10^{155}$ | $8 \times 10^{26}$ |

Birthday attack:  
\* First and second pre-image attack  
\* collision attack



cas MD5: blocs de 64b → birthday attack  $2^{64} < 2^{80} \Rightarrow$  non sécurisé

trop rapide à calculer (150 m de hash/s ~ CPA:  $2^{23} \Rightarrow 2^{23} = 2^{23} = 1h8$ )

cas SHA-1: collision dans  $2^{60}$  appels (meilleur que MD5 mais pas assez - 160 bits de sortie)

cas SHA-2: sha-224, 256, 384, 512. Pas de vulnérabilité connue, faut juste éviter de l'implémenter en 31 tours

cas SHA-3: alternative (pas juste remplacement). Plus flexible (utilisable avec plusieurs algos cryptographiques)

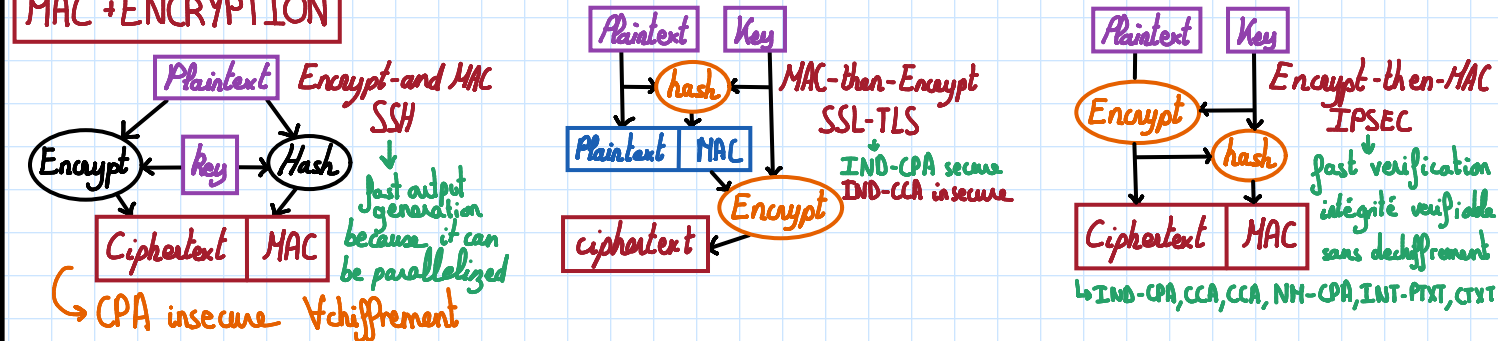
Limitation des fonctions de hachages: en cas de MITM sur un téléchargement, on envoie le matériel + son hash

absence d'authentification!

Authentification:

- ↳ AEAD (Authenticated Enc. with Associated Data) → l'attaquant peut faire un magicien et recateler le hash correspondant pour la victime
- ↳ contre-mesure: HMAC si le user/servant partagent un secret
- ↳ 1<sup>ère</sup> tentative: secret-prefix:  $(m, MAC = H(s||m))$  → un attaquant peut forger  $(m', h')$  tq  $m' = H(\text{padding}||K)$  et  $h' = \text{hash}(K)$
- ↳ 2<sup>ème</sup> tentative: secret-suffix:  $(m, MAC = H(m||s))$  → brute force
- ↳ vuln on second pre-imag att: si on trouve  $H(m_1) = H(m_2)$ , le MAC sera identique car l'état est le même avant d'ajouter s.
- ↳ vuln on weak collision: l'attaquant peut faire signer un message  $m_i$  légitime par récupérer un MAC valide ( $H(m_1) = H(m_2)$ )
- ↳ HMAC:  $HMAC(S_K, m) = H((S_K \oplus \text{opad}) || H(S_K \oplus \text{ipad}) || m)$  → repose sur pseudo aléatoire, donc on peut utiliser hachage faible (md5, sha-1), pas besoin de résistance aux collisions
- ↳ serveur malicieux (cas MD5): on note  $h_0 = h(\text{padding}||m_1) = h(\text{padding}||m_2)$  sont en collisions. que se passe-t-il  $S_K = p \oplus \text{ipad}$ ?
- ↳  $HMAC(S_K, m_1) = MD5(S_K \oplus \text{opad} || h_0) = HMAC(S_K, m_2)$  → problématique

MAC + ENCRYPTION



Rainbow Tables: calcul de condensat + réduction massivement sur beaucoup d'itération

- ↳ on stocke uniquement input + résultat (pas les condensats intermédiaires)
- ex: bonjour hash, reduce, gkp hash, reduce, jmb0 ... rbo → on stocke "bonjour; jmb0"
- si on applique 5 fois la boucle par exemple et qu'on cherche à décoder "gkp", il suffit d'itérer maximum 4 fois et de comparer les condensats avec les outputs de la rainbow table pour trouver l'antécédent
- ↳ mitigation: le SEL!!!
- on va concaténer le sel et l'élément à hasher et hasher le tout
- ↳ le sel est un secret (il doit être conservé de manière sécurisée)

Proof of Work (Hashcash): pour éviter les DOS → le début du hash doit être composé d'un padding nul

ce qui nécessite de nombreux essais → bitcoin

Key Derivation: L'objectif est de transformer un secret biaisé en clé cryptographique robuste

→ il faut utiliser la sortie uniforme des fonctions de hachage

## IV ALEAS, PSEUDO-ALEAS, CHIFFREMENTS À FLOTS

Le chiffrement à flot: on fait une suite de OTP, on veut chiffrer un message de taille quelconque

↳ on XOR avec un flux aléatoire

l'intérêt est de chiffrer un flux de données: typiquement les télécommunications (Wifi, Bluetooth, GSM)

- ↳ 1<sup>er</sup> problème: comment générer une clé de la même taille que le message → utiliser un générateur d'alea
- ↳ 2<sup>ème</sup> problème: comment échanger cette clé avec un interlocuteur → mécanisme d'échange de clé

générateurs d'aleas → pur: provient d'un phénomène physique ou quantique

pseudo-aléatoire: seed (grains) + PRNG (algo déterministe pour mélanger)

↳ il ne faut pas trop réutiliser la seed

PRNG: heure actuelle (insecure), compteur de cycle de processeurs, nombre de processeurs, déplacement sautés, générateurs matériels

LFSR: Linear Feedback Shift Register

↳ de base une seed

output → 3 XOR (simple à implémenter mais avoir à casser)

↳ algo de Berlekamp-Massey: 2 fois la taille de l'état interne suffit à le casser

LFSR - Amélioration:  $g(LFSR_1, LFSR_2, LFSR_3, \dots)$  = output (j non linéaire)

↳ LFSR filtré:  $g(X_1, X_2, \dots, X_n)$  = output avec  $g$  un filtre et  $(X_i)$  l'état du LFSR

LFSR - les ratés: DVD: CSS (2 LFSR combinés), GSM: AS/4 (3 LFSR), Bluetooth: EO (4 LFSR) → tous cassés

↳ UMTS/LTE: snow2/snow3g ✓

chiffrements à flot:

- ↳ RC4: WEP, TKIP, HTTPS
- ↳ si il ne faut pas trop réutiliser la clé:  $(PRNG(S_K) \oplus m_1) \oplus (PRNG(S_K) \oplus m_2) = m_1 \oplus m_2$
- ↳ chiffrement/déchiffrement à la volée
- ↳ pour éviter le délai on peut pré-générer le pseudo-alea
- ↳ c'est sûr sans résoudre de la sécurité du PRNG

## V ECHANGE DE CLÉS

1<sup>ère</sup> solution: tier de confiance (on file la clé à un tier qui s'occupe de la communiquer à notre destinataire)

↳ il faut souvent passer par le tier de confiance (bcp de clés) + il faut lui faire confiance

Diffie-Hellman (1976): polynomial pour les 2 parties, exponentiel pour un attaquant.

on se place dans un corp fini, p premier ( $> 1024$  bits), q premier tq  $p = 1 + Kq$  ( $> 1000$  bits)

on peut trouver g EN générateur tq:  $g^q = g[p]$  et  $\forall i \in \mathbb{Z}_q, i \neq 0 \Rightarrow g^i[p] \neq g^j[p]$

Alia: a

Bob: b

un attaquant peut quand même faire  $S_K = P_K[a]$  et  $S_K = P_K[b]$

un MITM

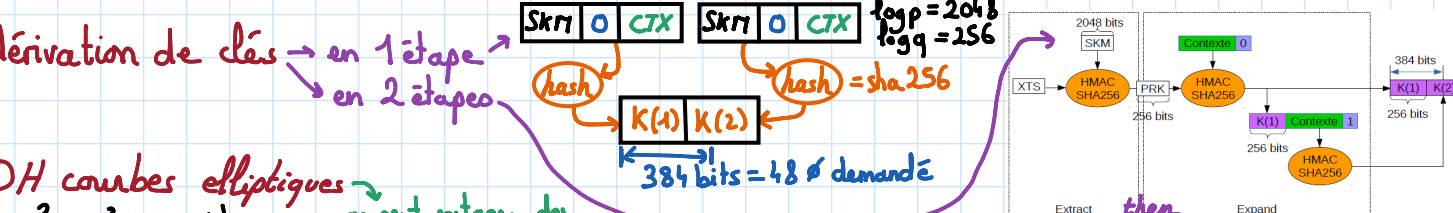
il faut de la sécurité persistante



**Alia:**  $a \leftarrow U_q$   
 $p_a = g^a[p]$   
 $s_k = p_a[b]$   
**Bob:**  $b \leftarrow U_q$   
 $p_b = g^b[p]$   
 $s_k = p_b[a]$   
clé statique  
clé éphémère  
suite des échanges avec  $S = s_k || s_k \rightarrow$  ne pas utiliser pour chiffrer  $\rightarrow$  dériver une clé (KDF)  $\rightarrow$  alia uniforme

**Pejfect Forward Secrecy (PFS) - compromission de clé**  
 $\rightarrow$  cette fois-ci, si Osama hack le serveur, il récupère  $s_k$  mais ne peut rien déchiffrer.  
 $\rightarrow$  le premier échange demande quand même une authentification pour éviter les MITM

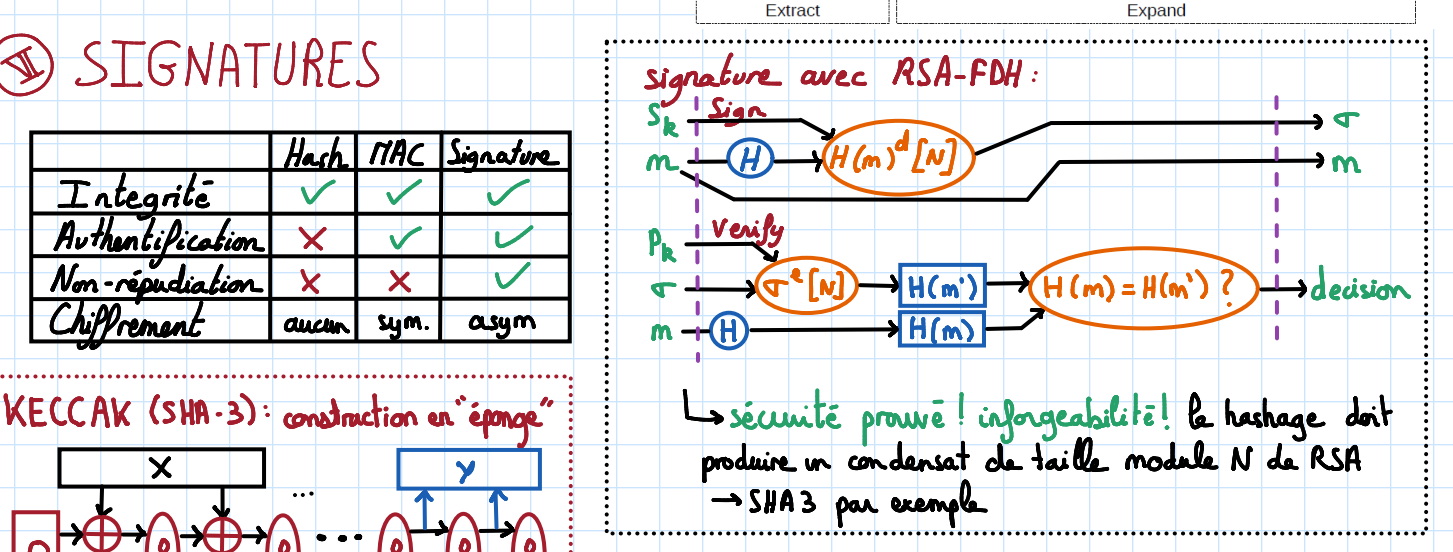
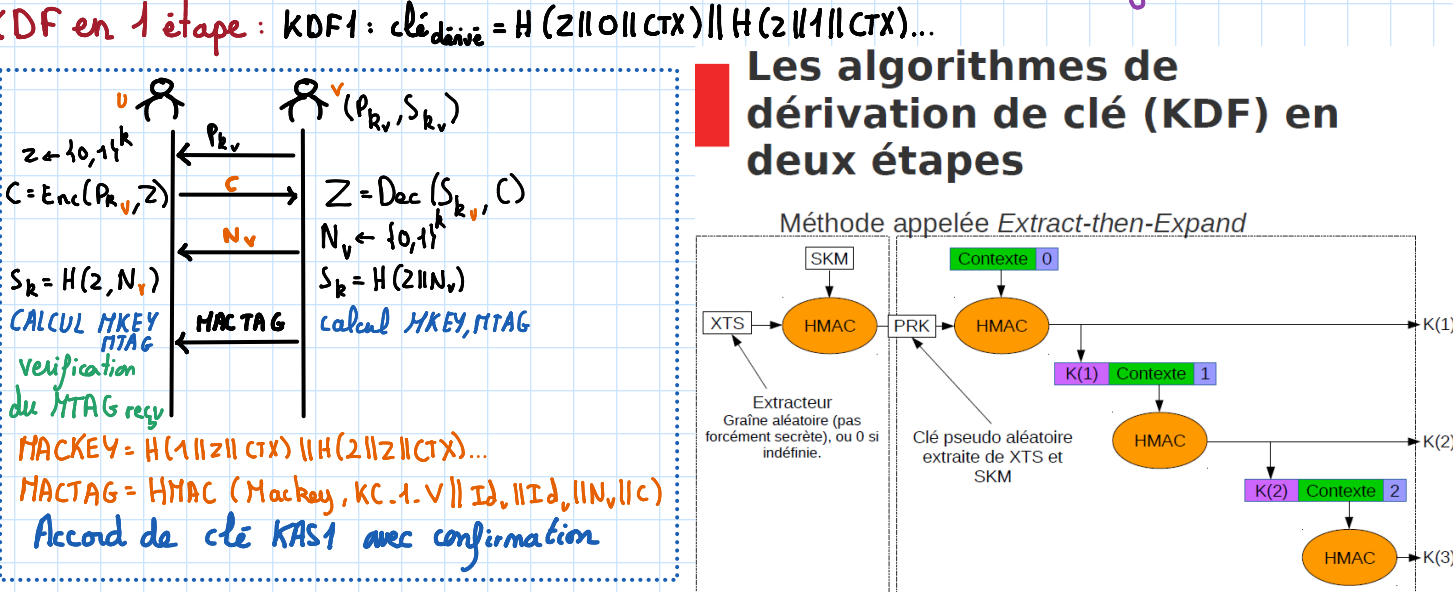
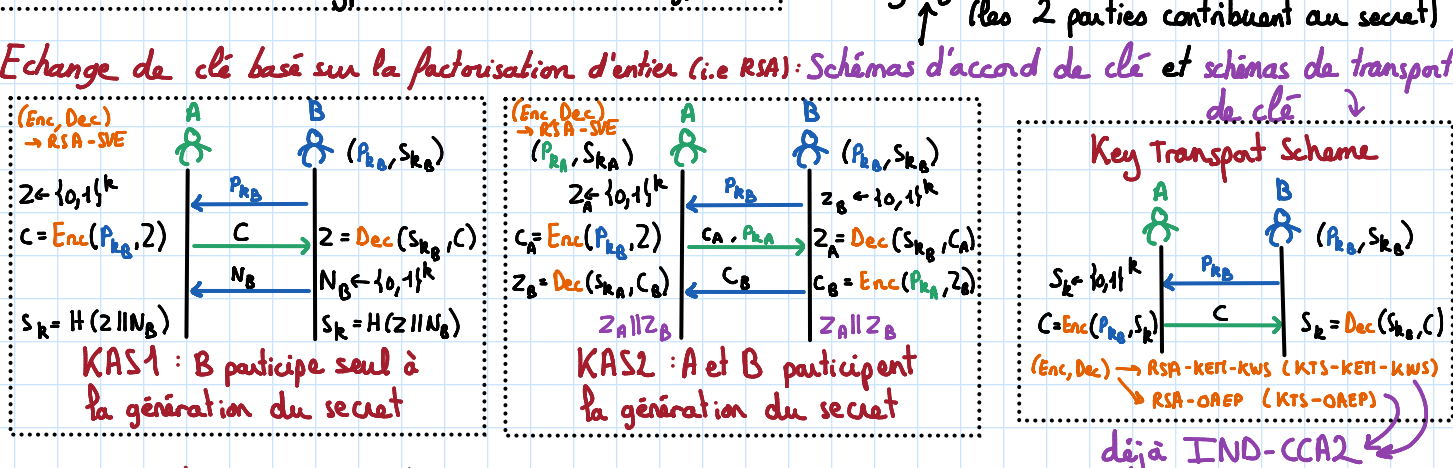
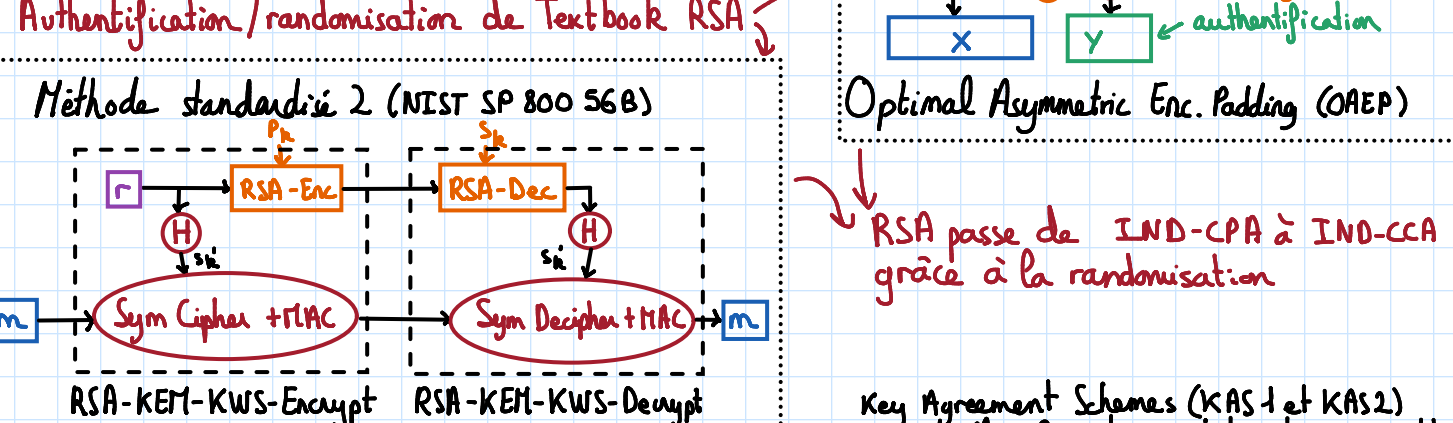
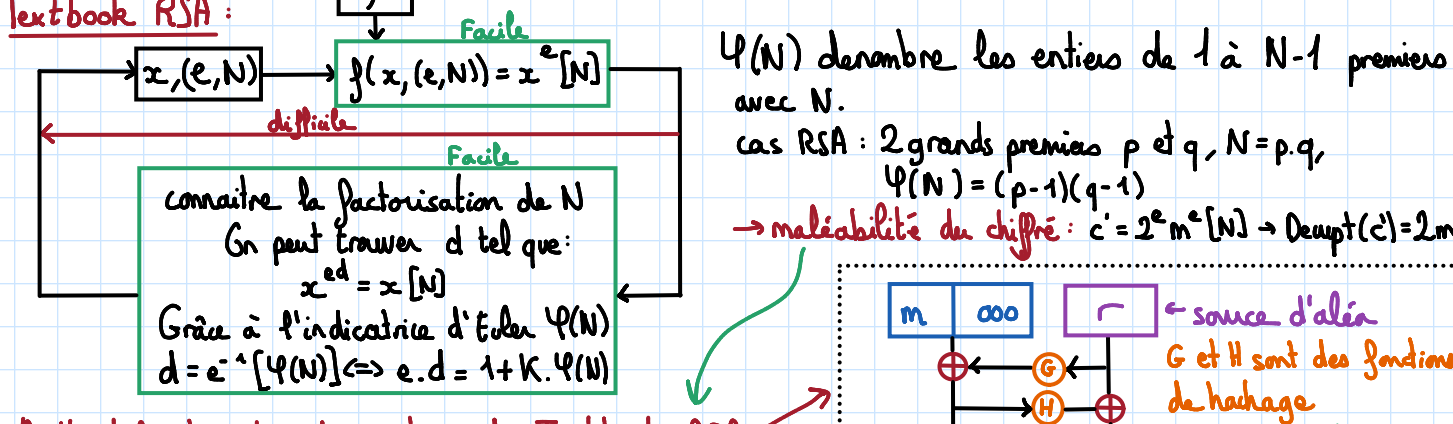
**Sécurité de Diffie-Hellman:** connaissant  $(g, p, g^a, g^b)$ , trouver  $x = g^{ab}[p]$  difficile (computational DH: CDH)  
casser le logarithme discret  $\Rightarrow$  casser DH (l'inverse n'est pas prouvé)  
niveau de sécurité: 80 bits  $\rightarrow (\log p, \log q) = (1024, 160), 112 \rightarrow (2048, 224), 112 \rightarrow (2048, 256), > 112$  possible mais casse fini non recommandé



**DH courbes elliptiques**  
 $-y^2 = x^3 + ax + b$   
on peut partager des clés symétriques de sécurité 256 bits!  
- P+Q = R (addition de points)  
① intersection courbe et droite PQ  
② droite  $y = xs + t, s = \frac{y_a - y_p}{x_a - x_p}, t = \frac{x_p y_a - x_a y_p}{x_a - x_p}$   
③  $y_R = s^2 - y_p - y_a, x_R = s(y_p - y_a) - x_p$   
- doublement de P: P+P=R  
① intersection courbe/tangente  
②  $y = xs + t, s = \frac{3x_p^2 + a}{2y_p}, t = -(\frac{3x_p^2 + a}{2y_p} x_p + y_p)$   
③  $y_R = s^2 - 2y_p, x_R = s(y_p - y_R) - x_p$

**comparaison**  
 $\rightarrow$  corps finis: 1 multiplication de point = 1 mult. entière. Pour 128 bits de sécu:  $L_p[1/3, 1.92], \log p = 3076$   
 $\rightarrow$  courbes elliptiques: 1 mult. de point = 14 mult. entière. Pour 128 bits de sécu:  $L_p[1, 0.5], \log p = 256$

chiffrements symétriques sont construits sur des fonctions à Trappe  
sachant  $f(x, p_k)$  et  $P_k$ , trouver  $x$  est difficile (temps exponentiel)  
sachant  $f(x, p_k)$  et  $S_k$ , trouver  $x$  est facile (temps polynomial)



**Ex: serveurs:** lorsqu'on déploie un patch on le signe, il faut alors truster le certificat Racine et hiérarchiser les certificats (racine, signe, pr qui signe, insa, pr...)  $\rightarrow$  si l'un des signataire est compromis: MITM.

**Web of Trust:** Les utilisateurs s'inter-signent. Si tu es authentifié par assez de noeuds, on peut te faire confiance. Le 4u devient un noeud.

**CRL:** Liste de révocation: on peut invalider un certificat valide cryptographiquement en l'ajoutant à une blacklist.

**identité de Frobenius:** dans  $\mathbb{F}_p(\mathbb{Z}/p\mathbb{Z})$  avec  $p$  premier:  $(a+b)^p = a^p + b^p$   
 $\rightarrow$  ex:  $x^3 + 1 = (x+1)^3$  dans  $\mathbb{F}_2$  car  $(x+1)^3 = [(x+1)^2]^2$

$\mathbb{Z}/p\mathbb{Z}$  avec  $p$  premier =  $\mathbb{F}_p$  est un anneau possédant  $p$  éléments  $\rightarrow$   $\begin{matrix} a+b \\ a \cdot b \end{matrix}$

$G$  non vide muni de la loi de composition  $+$ :  
 $V(x, y) \in G^2, x+y \in G$  (ensemble fermé)  
 $Vx \in G, 30 \in G$  tq  $0+x = x+0 = x$  (élément neutre)  
 $Vx \in G, \exists x^{-1} \in G$  tq  $x+x^{-1} = 0$  (inverse)  
 $V(a, b, c) \in G^3, a+(b+c) = (a+b)+c$  (associativité)

$\Rightarrow G$  est un groupe, on le note  $(G, +)$   
 $\rightarrow$  si  $+$  est commutatif  $\Rightarrow$  le groupe est abélien

$A$  non vide muni de  $+$  et  $\times$   
 $(A, +)$  abélien de neutre  $0$   
la loi  $\times$  est associative  
 $V(a, b, c) \in A, (a+b) \times c = a \times c + b \times c$   
 $c \times (a+b) = c \times a + c \times b$

$\Rightarrow A$  est un anneau, noté  $(A, +, \times)$   
 $\rightarrow$  dit commutatif si  $\times$  commutative  
 $\rightarrow$  dit unitaire si  $\times$  possède un neutre  $1 \in A$  (appelé unité de l'anneau)

**anneau unitaire**  $(K^*, \times) \Rightarrow K$  est un corps  
(il faut que tous les éléments d'un corps aient un inverse sauf le nul)

**TD & Annales**

**ephemeral**  $\rightarrow$  sinon pas de PFS  
 $\rightarrow$  en 1.3 ça précise pas  $\rightarrow$  PFS ✓

chiffre suite EC/DHE ou DHE pour garantir la PFS  $\rightarrow$  pas les "RSA..."  
Niveau de sécurité à 128 bits min recommandé par l'ANSSI d'ici 2030.

$a^n - b^n = (a-b) \sum_{k=0}^{n-1} a^{n-1-k} b^k$   
 $\log_a b = b$

**RSA (key, security):** (1024, 80), (2048, 112), (4096, 128)

d'ici 2030, min 128 pour AES

**MAC = "certificat + hash" (en gros)  $\rightarrow$  intégrité + authentification**  
 $\rightarrow$  HMAC  $\rightarrow$  MAC basé sur un hash  
 $\rightarrow$  on envoie  $H(\text{secret} || \text{message})$ , le recepateur possède le secret et vérifie en recalculant le hash  $(s, m) \rightarrow$  attaquable par extension

**non-répudiation:** signature (asymétrique, car signature symétrique pour prouver la signature faut filer la (seule) clé)

**ANKI SIDE CHANNEL**  $\rightarrow$  par conso énergétique, champ élecnaq, temps d'accès au cache  
 $\rightarrow$  attaque parchi (Spectre, ZenBleed, Rowhammer), attaque Archi (Buffer Overflow, ROP)  
 $\rightarrow$  attaque DPA (Différentielle):  $c = a \oplus b \rightarrow$  on fait 1000 traces avec  $b$  qui varient  
 $\rightarrow$   $c$  était nul donc bitflip proportionnelles aux nombres de '1'  
 $\rightarrow$  on estime  $a$  et on simule avec le nb de bitflip sur  $b$   
 $\rightarrow$  on trie les traces (simulations pour  $b=0, 1, 2, 3, \dots$ )  
 $\rightarrow$  si on a correctement estimé  $a$ , la moyenne des traces hautes devraient être plus haute que la moyenne basse. Si on s'est trompé la trie est aléatoire: moyenne équivalente des 2 groupes  
 $\rightarrow$  attaque CPA (Correlation Power Analysis): ex: 50 000 clairs  $\oplus$  et les chiffre avec AES  
 $\rightarrow$  cas iii: pas de mixcolumn sur dernier round (ex)  
 $\rightarrow$  en théorie le mixcolumn crée de la diffusion et le shiftRows propage la diffusion  
 $\rightarrow$  on suppose la clé entre 0 et 255, on XOR avec la 1<sup>re</sup> octet de la matrice du chiffre  
 $\rightarrow$  on inverse AES (on fait les étapes à l'envers) \*  
 $\rightarrow$  on obtient le bloc avant ET après le dernier tour  
 $\rightarrow$  on regarde la distance de hamming entre  $B^n$  et  $B^{n-1}$   
 $\rightarrow$  on regarde la distance de hamming entre  $B^n$  et  $B^{n-1}$   
 $\rightarrow$  on rejette l'opération pour les 50K chiffres  
 $\rightarrow$  on a les traces de conso et on regarde les corrélations entre les simulations et la courbe réel  $\rightarrow$  si on a bien estimé la clé la corrélation sera bonne  
 $\rightarrow$  on va faire tout et chercher à maximiser la corrélation

$b(x) = v(x) + a(x) w(x) [x^3 + 1] \rightarrow a(x) = \frac{b(x) - v(x)}{w(x)} [x^3 + 1]$   
 $\rightarrow$  inversible  $\Leftrightarrow w(x) \neq 0$  et irréductible ou non divisible par  $x^3 + 1$

