

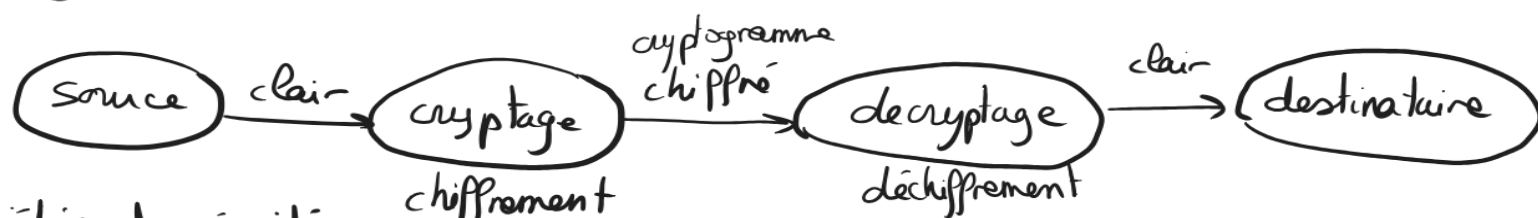
I Introduction/historique

1 exam + 1 CTF → crypto

1 exam → crypto post-q

cryptanalyse → analyser des messages

cryptologie → science du secret



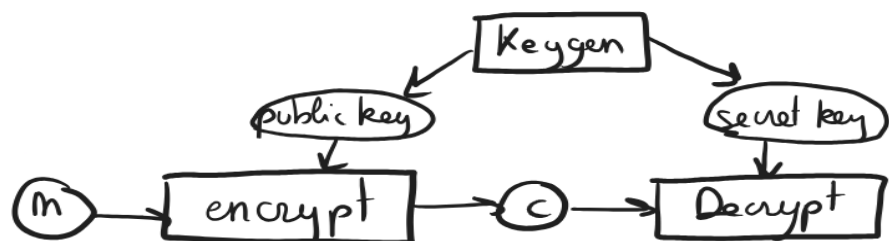
propriétés de sécurité :

- confidentialité → impossible de déduire de l'info à partir du chiffré sur le clair
- intégrité → si le chiffré est modifié, le destinataire peut le détecter (hachage par ex)
- authentification → toute usurpation peut être détectée
- non-répudiation → si quelqu'un envoie un message, il ne peut pas nier qu'il a émis ce message

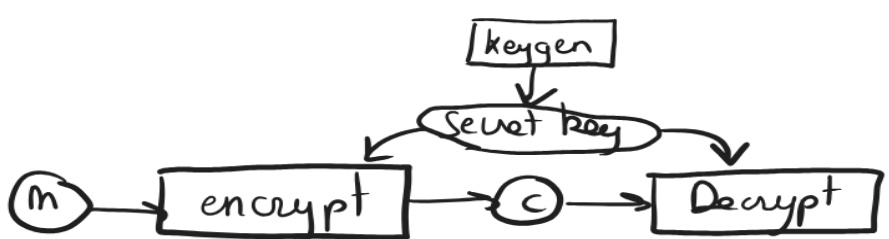
canaux non sûr, sûr, confidentiel et authentique : cf. p. 5 cours intro.

NIST/IETF ✓

ISO → PTDR c'est naze



chiffrement asymétrique



chiffrement symétrique
pas d'authentification

on note $x \stackrel{\mathcal{X}}{\leftarrow} S$ ou $x \leftarrow (S, \mathcal{X})$ le tirage d'un élément de S selon la distribution $\mathcal{X}$

- chiffrement : $c \leftarrow \text{Enc}(s_k, m)$
- déchiffrement : $m \leftarrow \text{Dec}(s_k, c)$
- consistance : $\forall m, s_k : \text{Dec}(s_k, \text{Enc}(s_k, m)) = \{m\}$
- sécurités

propriétés
de
chiffrement

chiffrement parfaitement sûr :

$$\hookrightarrow H(C|M) = H(C)$$

$\hookrightarrow |K| \geq |M|$ → l'attaque de l'espace de clés est au moins aussi grand que l'attaque de l'espace de message

• $\oplus$ addition modulo 2

$$\hookrightarrow x \oplus y \oplus x = y$$

one time pad : pour un chiffré : tout clair est possible

$$\text{clair} \oplus \text{clé} = \text{chiffré} \rightarrow \text{chiffré} \oplus \text{clé} = \text{clair}$$

• chiffrement de Vernam = one time pad

maléabilité : $c' = c \oplus x$ est un chiffré valide du message $m' = m \oplus x$

$\hookrightarrow$ faire m XOR sur le chiffré, c'est faire un XOR sur le clair dès qu'il sera décodé

$$\text{clé non réutilisable : } c_1 + c_2 = (m_1 \oplus k) \oplus (m_2 \oplus k) = m_1 \oplus m_2$$

Question

Alice et Bob se mettent d'accord sur une clé et souhaitent s'envoyer des requêtes de type "Y" (yes) et "N" (no) en chiffrant leur message avec le chiffre de Vernam. Alice, après déchiffrement, récupère le message "N". Quelle confiance peut-elle avoir en ce message ?

Reponse

Assez faible. En cas d'interception par un attaquant, la propriété de maléabilité peut lui permettre d'interchanger la valeur du message :

$$c' = c \oplus (Y \oplus N)$$

→ si on connaît le format, encore une limitation

II Chiffrements Symétriques

• Substitution → chiffrement par décalage → ex : Code de César

→ attaquable fréquemment

• Substitution poly-alphabétique → Vigenère → Vernam est un cas particulier de Vigenère.

• Substitution polygraphique → Chiffre de Playfair

→ Chiffre de Hill → dans AES btw.

$$\begin{pmatrix} C_k \\ C_{k+1} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} P_k \\ P_{k+1} \end{pmatrix} [26] \quad \text{avec } P_k \text{ le } k^{\text{ème}} \text{ caractère du message, } C_k \text{ le } k^{\text{ème}} \text{ du chiffré}$$

$$\text{nb de solution } 157 \cdot 248 \approx 2^{17,26} \Rightarrow \lambda = 17,26 \Rightarrow \text{niveau de sécurité faible}$$

• Permutation (changement d'ordre des symboles)

→ Permutation à croix grecques par exemple

→ Permutation à tableau complet

Auguste Kerckhoffs

- ↳ le cryptosystème doit être inviolable en pratique même s'il ne l'est pas dans les faits
- ↳ aucun secret dans le cryptosystème (car connaître le cryptosystème ne doit pas le compromettre)
- Chiffre ADFGVX : substitution + permutation.
- Machine de Lucifer (IBM/1960)

→ renforcement de la sécurité grâce à une opération non linéaire (S-Box par exemple)

AES → post quantum-proof avec une clé de taille suffisante

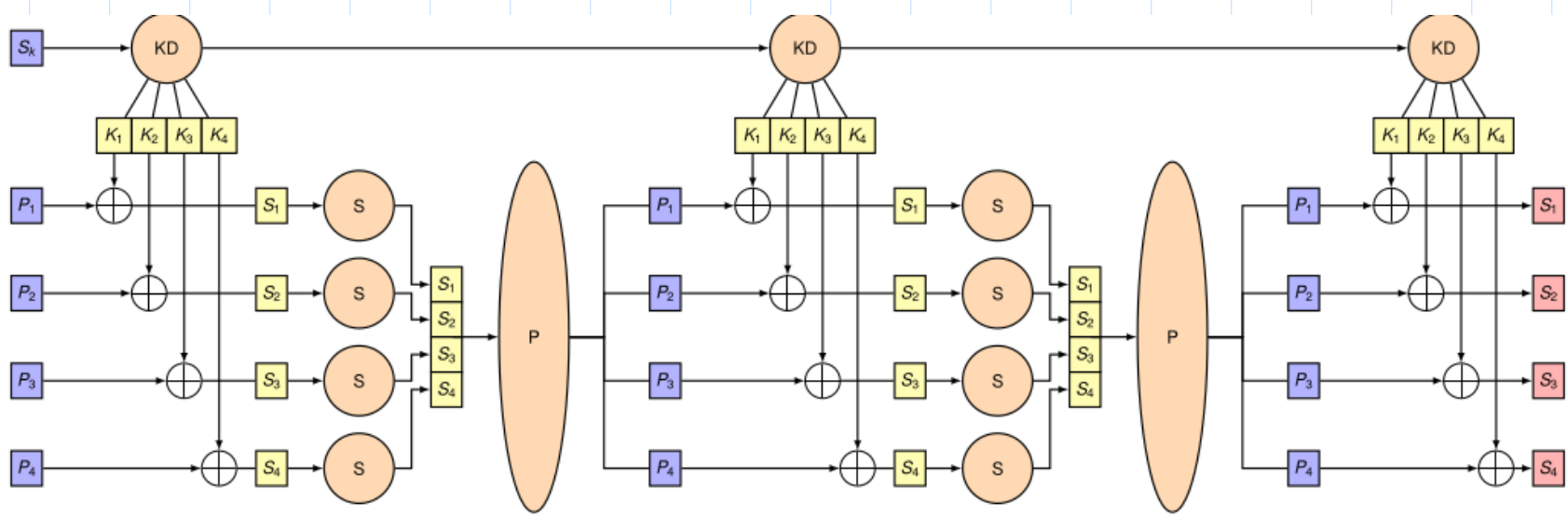
↳ revoir les slides pour comprendre l'algo

effet d'avalanche : dans un chiffrement avec bonne diffusion, l'inversion d'un seul bit de l'entrée doit changer chaque bit en sortie avec une proba de 0,5 (critère d'avalanche strict)

→ AES : toujours 128 bits (AES-128, AES-192, AES-256 correspond à la taille de la clé et niveau de sécurité)

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

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



S-Box : non linéaire, substitue, confusion

P-Box : linéaire, mixe, diffusion

effet d'avalanche : 1 bit flip induit n bit flip (si "avalanche stricte" 50% des bits)

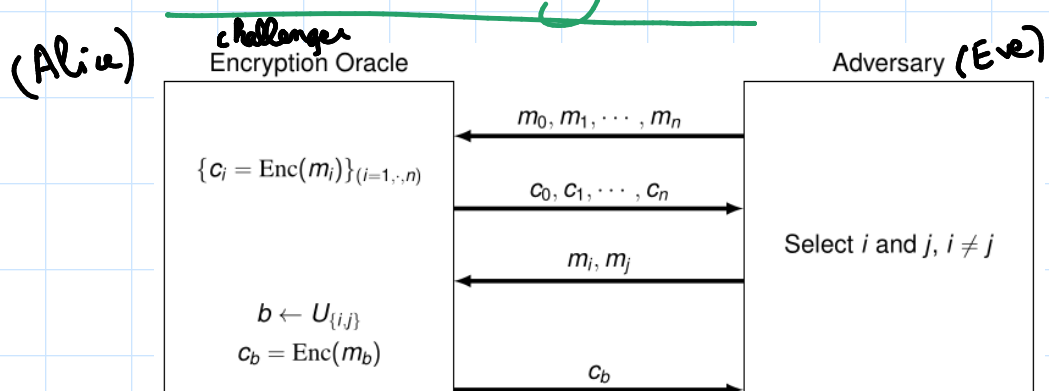
chiffrer des messages de grandes tailles : mode d'opération

- ↳ ECB → raze de fou → si on a le même bloc en entrée on aura la même sortie (car déterministe)
- ↳ CBC → moyen bien → on chiffre un bloc avec la clé et le chiffré du bloc précédent.
- ↳ GCM → peu vulnérable → chiffrement de bloc pour générer un flux de clés + chiffrement de Vernam avec ce flux de clés.

evaluate security?

- IND-KPA (Known plain-text attack) → $(m_i, \text{Enc}(m_i))$ known
- IND-CPA (chosen plain-text —) → select m_i and asks to encrypt m_i
- IND-CCA → can also select c_i and ask to decipher

IND-KPA game

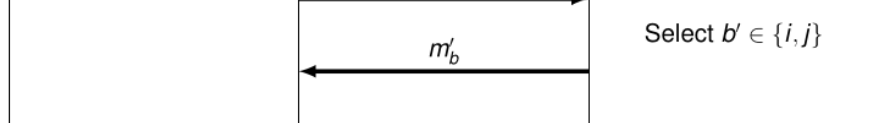


Win condition

- Adversary wins the game if: $\Pr[b = m_0] > 1/2$.
- If $\Pr[b = m_0] = 1/2$, then adversary can only guess randomly which message has been encrypted.
- Advantage: $\mathcal{A}_{CPA} = |\Pr[b = m_0] - 1/2| = \epsilon$

Notion of negligible advantage

- For key length k ;
- For Advantage $\mathcal{A}_{CPA} = |\Pr[b = m_0] - 1/2| = \epsilon(k)$;
- Adversary has negligible advantage if $\epsilon(k) < \frac{1}{p(k)}$ for all k after given k .



→ ECB mode of operation with CPA: how to win the game?

$m_i = [\text{hello}][\text{world}]$

$m_j = [\text{hello}][\text{hello}]$

$\Rightarrow \text{Enc}(m_i) = [c_0][c_1], \text{Enc}(m_j) = [c_0][c_0]$

can it differ per block et algo déterministe

⚠ d'où ECB cagare! $\mathcal{A}_{\text{CPA}} = 1/2$

→ CBC mode with CPA: how to win the game?

on choisit le vecteur d'initialisation

under free nonce

$m_i = [\text{Hello}] \quad m_j = [\text{World}]$
 $\text{Enc}(m_i) = [c_i], \text{Enc}(m_j) = [c_j]$

then choose [Hello] and [World] as challenge

globalement il algo déterministe est sensible aux attaques par replay.

random, secret, but repeated

Select m_i, m_j before challenge and request $c_i = \text{Enc}(m_i), c_j = \text{Enc}(m_j)$

nonce should not be used twice

randomized, public but predictable

pas secure non plus, j'ai la plume de noter mais pas secure.

Combien de bloc peut-on chiffrer avant de compromettre la clé?

CBC - theorem

For any length $L > 0$:

If PRP E is semantically secure over (K, X) , then E used in CBC mode (E_{CBC}) is semantically secure under CPA over (K, X^L, X^{L+1}) .

For adversary making q -query, then:

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

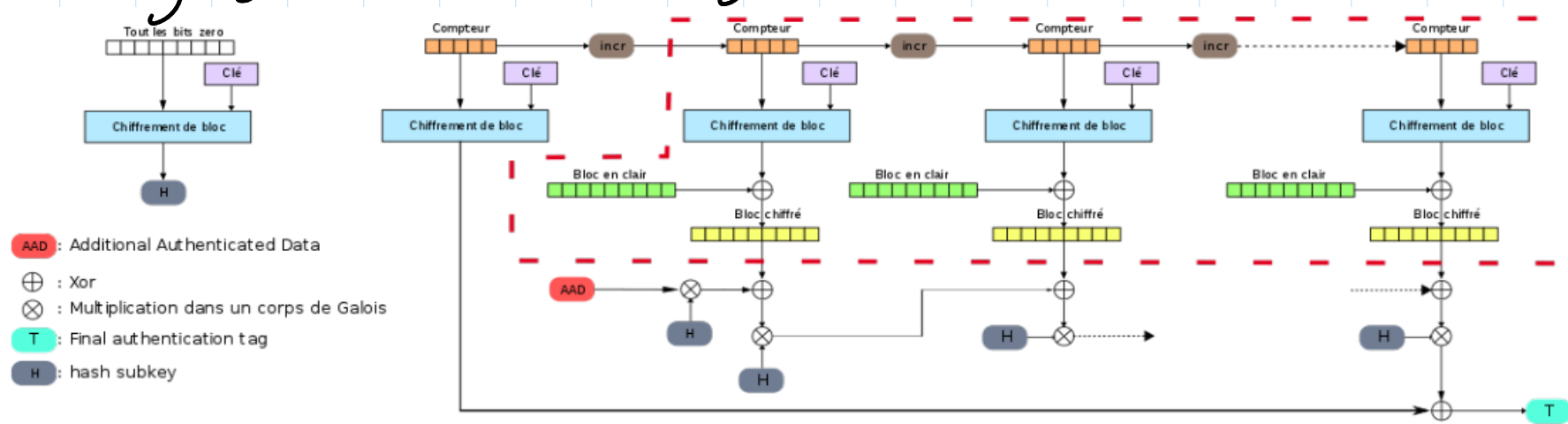
Where $|X|$ is the number of outputs possible for the permutation and L the maximum number of blocks per message.

case of AES: (size: 128bits, target advantage: $2^{-80} = q^2 L^2 / |X|$)
 $\Rightarrow qL = \sqrt{2^{-80+128}} = 2^{24}$ encrypted blocks

→ GCM

→ AES produit un effet d'avalanche donc la clé va suivre une distribution uniforme

→ on y ajoute de l'authentification



III Integrity and Authentication: Hash

desired properties:

- shortest check code
- robustness against bit flip → effet d'avalanche
- avoid same hash from 2 different inputs

cryptographic hash function: - finding m such as $H(m) = H(m')$ is hard knowing $H(m)$

cryptographic hash functions:

- finding m_2 such as $H(m_1) = H(m_2)$ is hard knowing $H(m_1)$
- finding m_1 and m_2 such as $H(m_1) = H(m_2)$ is hard

first pre-image resistance
weak collision resistance
strong collision resistance

condensat = hash on empreinte

Rainbow Table

- calcul de condensat + reduction massivement sur bcp d'itération
- on stocke uniquement input + résultat (pas les condensats intermédiaire)

rainbow $\xrightarrow{\text{Hash}} \xrightarrow{\text{Reduc}} \text{gRp} \xrightarrow{\text{Hash}} \xrightarrow{\text{Reduc}} \text{jmbO} \rightarrow \dots \rightarrow \text{rbO}$

→ si on applique 5 fois la boucle par exemple et qu'on cherche à décoder "gRp" on sait qu'en bouclant 5 fois maximum (Hash; Reduc) on devrait matcher un output de la Rainbow Table et appliquer Hash; Reduc par retrouver un précédent dont l'empreinte serait similaire

Rainbow Table:
rainbow/rbO
...

→ counter: SALT !!!



→ on va concatener le sel et l'élément et tout hasher
→ le sel doit être tenu secret sur le serveur

Proof of Work: avoid DOS

Key Derivation:

chiffrements homomorphes: si on additionne sur le chiffré, ça additionne sur le clair (ex si homomorphe sur l'addition). chiffrant car 1bit de clair → 40kb de chiffré!!

AES en mode CTR: style à flux mais en réalité on génère les clés par bloc de 128
→ dans TLS 1.3 → réel chiffrement à blocs: Salsa 20

chiffrements à blocs: même problème que one time pad → la clé secrète ne doit pas être trop réutilisée car si algo déterministe et pas de seed (vecteur d'initialisation), on aura le même output.

④ Echanges de clés

Diffie-Hellman (1976):

corps fini - $p > 1024$ bits premier
- $q > 100$ bits premier tq $p = 1 + k \cdot q$
 $\mathbb{Z}_p$ = ensemble des entiers modulo p .

→ difficulté polynomiale mais exponentiel pour les attaquants.

... il faut générer les clés car sinon l'attaquant peut générer

Man in the middle (MITM) : il faudrait partager les clés en secret.
 2 secrets partagés (1 avec Alice et 1 avec Bob)

mutual TLS : le client ET le serveur ont des certificats

sécurité persistante : se mettre d'accord sur p, q et g

computational Diffie-Hellman

Diffie Helman : connaissant (g, p, g^a, g^b) déduire $x = g^{ab} [p] \rightarrow$ problème CDH difficile

Complexité du logarithme discret : $L_p[\alpha, c] = 2^{(c + o(1))(\log p)^\alpha (\log \log p)^{(1-\alpha)}}$
 algo polynomial au exponentiel $\downarrow \alpha=0$
 $L_p[1, c] = 2^{(c + o(1)) \log p}$
 $L_p[0, c] = (\log p)^{(c + o(1))}$
 } $\log$ ici c'est base 2

$\rightarrow$ meilleure méthode : crible algébrique : $L_p[1/3, 1.92]$

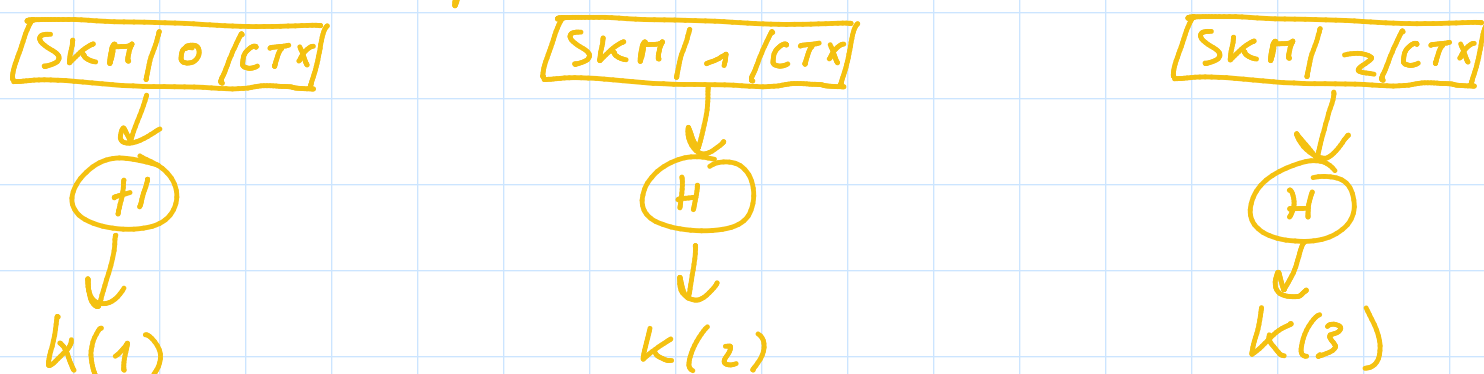
$\rightarrow$ si on souhaite doubler la complexité (doubler l'exposant) $\leadsto$ alors on multiplie $\log p$ par $2^3 = 8$

~~il n'y a~~ aucune preuve de la possibilité de la contraindre jusqu'à lors

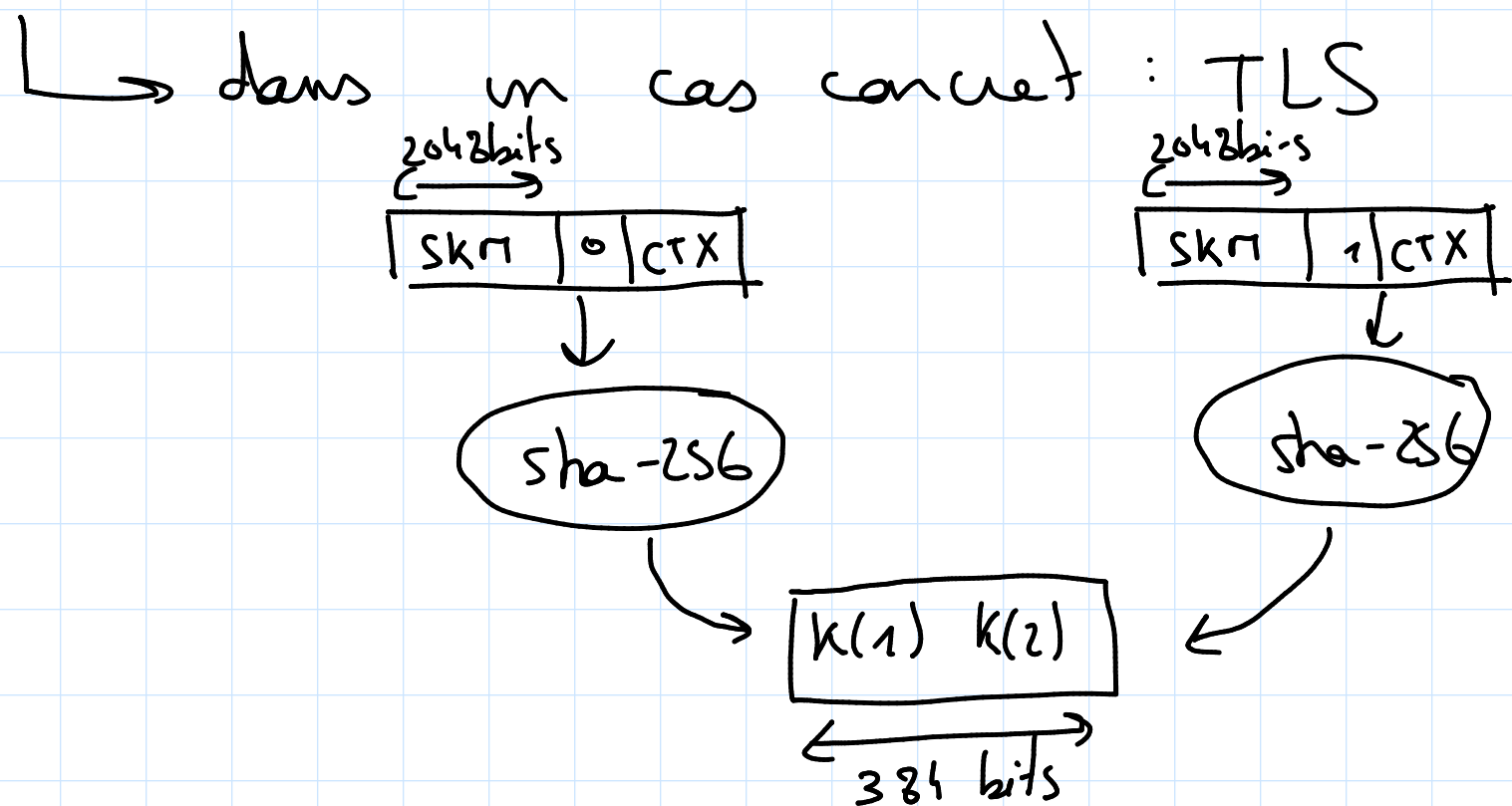
casser logarithme discret $\Rightarrow$ casser DH

logarithme discret : lien entre $\log p / \log q$ / niveau de sécurité trouvable sur la RFC 5114

Dérivation de clés en 1 étape (CAS 1)



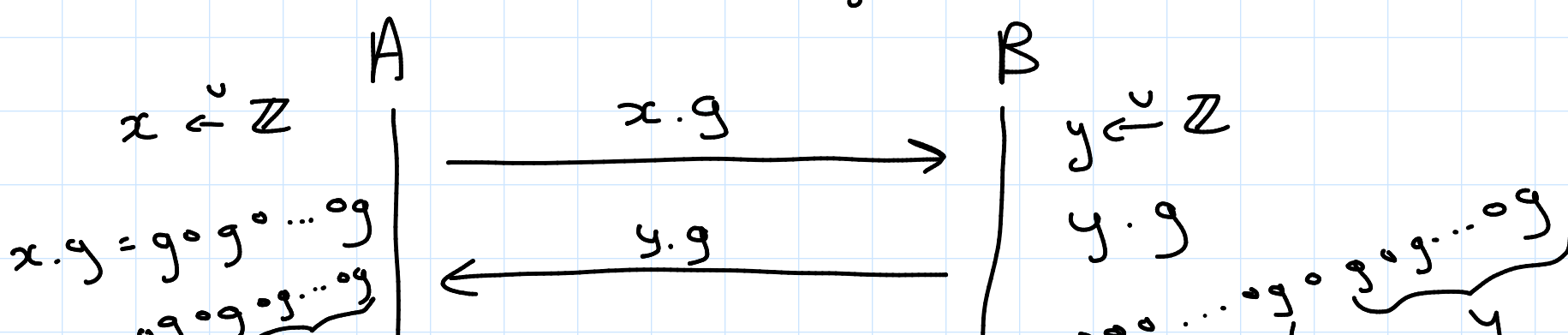
effet d'avalanche.
SKM : Secret Key Material
= secret partagé



Dérivation en 2 étapes (CAS 2) $\leadsto$ on s'étale pas dessus

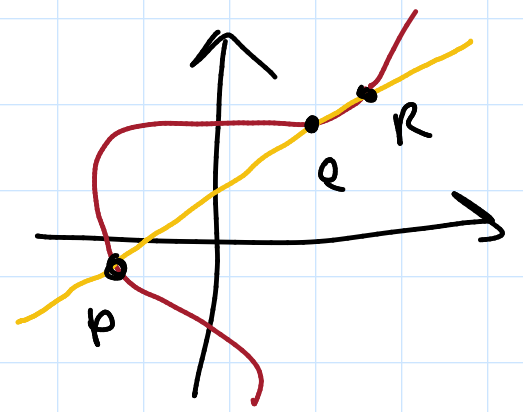
Diffie - Helman n'est pas restreint aux corps finis

$\rightarrow$ ex : G avec opération $\circ$, g générant un sous groupe G' de G de taille q .
 alors connaissant $\underbrace{g \circ g \circ \dots \circ g}_{x \text{ fois}}$, trouver x est difficile



Diffie-Hellman sur courbes elliptiques → avant c'était sur corps finis

- $y^2 = x^3 + ax + b$
- addition de points: $P + Q = R$
 - intersection courbes et droite PQ
 - droite: $y = xs + t$, $s = \frac{3x_p^2 + a}{2y_p}$, $t = y_p - \frac{(3x_p^2 + a)x}{2y_p}$
 - $y_R = s^2 - 2y_p$
 - $x_R = s(y_p - y_R) - x_p$



comparatif :

	corps finis	courbes elliptiques
par 1 multiplication de point	1 multiplication entière / point	14 multiplications entières / point
par 128 bits de sécurité	$L_p [1/3, 1.92] \Rightarrow \log p = 3076$	$L_p [1, 0.5] \Rightarrow \log p = 256$

Échange de clé non-interactif

Publication des clés sur un réseau social :

- A, B, C, D publient leur clé publique sur leur réseau social :
- $p_a = g^a [p]$, $p_b = g^b [p]$, $p_c = g^c [p]$, $p_d = g^d [p]$
- Supposons que **A** veuille communiquer avec **C**, combien d'échange faut-il avec C pour dériver une clé secrète ?
 - 0 : A récupère la clé publique de C et calcule p_c^a

Ce qu'il faut retenir (1)

- Diffie-Hellman (DH) est un mécanisme d'échange de clé dont la sécurité repose sur la complexité de calculer $g^{ab}[p]$ sachant (g, p, g^a, g^b) .
- Le Logarithme discret permet de casser DH, c'est même la meilleure méthode connue.
- Il existe deux grandes instantiations de DH :
 - Une instantiation sur les corps finis
 - Une instantiation sur les courbes elliptiques

Ce qu'il faut retenir (2)

- Sous réserve d'un choix judicieux de courbe elliptique, la version basée sur les courbes elliptiques possède un rapport sécurité/performance favorable.
- En particulier, la norme limite l'usage de DH sur les corps finis à un échange de clé symétrique de sécurité 112 bits.
- DH sur les courbes elliptiques peuvent attendre un partage de clé symétrique de sécurité 256 bits.

Ce qu'il faut retenir (3)

- Sans mécanisme supplémentaire, DH ne protège pas contre l'attaque du « Man-In-The-Middle », il faut prévoir une authentification de l'échange.
- Avec une mineure modification, DH permet

d'attendre une confidentialité persistante (Perfect Forward Secrecy).

• Pour se faire :

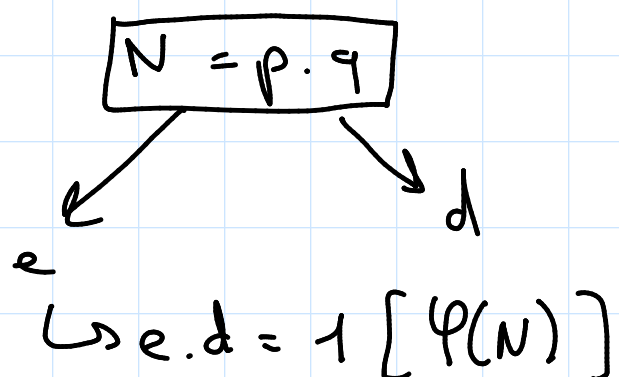
- un premier échange permet de se mettre d'accord sur une clé statique.
- Puis, pour chaque session, on génère un secret éphémère qui est ensuite détruit.

Diffie-Hellman n'est pas un chiffrement, RSA l'est

RSA

→ chiffrement asymétrique → fonctionne avec une fonction à trappe

↳ polynomiale à appliquer
↳ exponentiel pour récupérer l'antécédent
↳ sauf avec un secret !!



$$x^{ed} \equiv x \pmod{N}$$

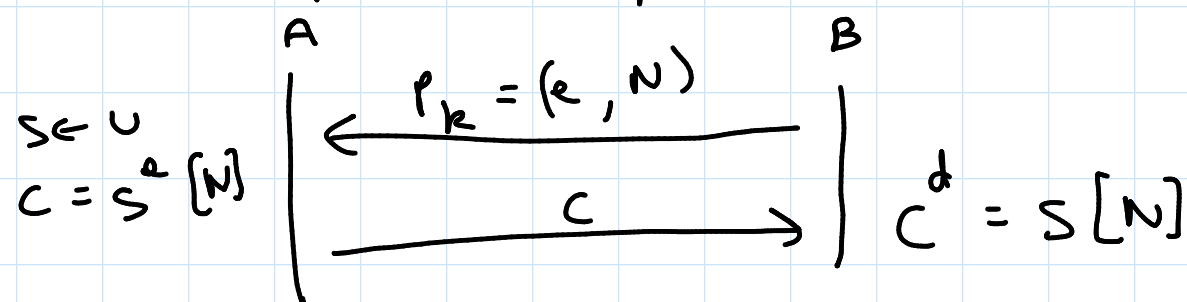
- $(P_k, S_k) \leftarrow \text{KeyGen}(1^\lambda)$

- $f : (x, P_k) \rightarrow f(x, P_k)$

- Sachant $f(x, P_k)$ et P_k , trouver x est **difficile** (en temps polynomial).

- Sachant $f(x, P_k)$ et S_k , trouver x est **facile**

→ limites de RSA : l'attaquant récupère $C = m^e \pmod{N}$, peut-il modifier le message dans C ?



→ pas homomorphe → ex : $C' = 2^e \cdot m^e \pmod{N} \rightarrow \text{Decrypt}(C') = 2 \cdot m$

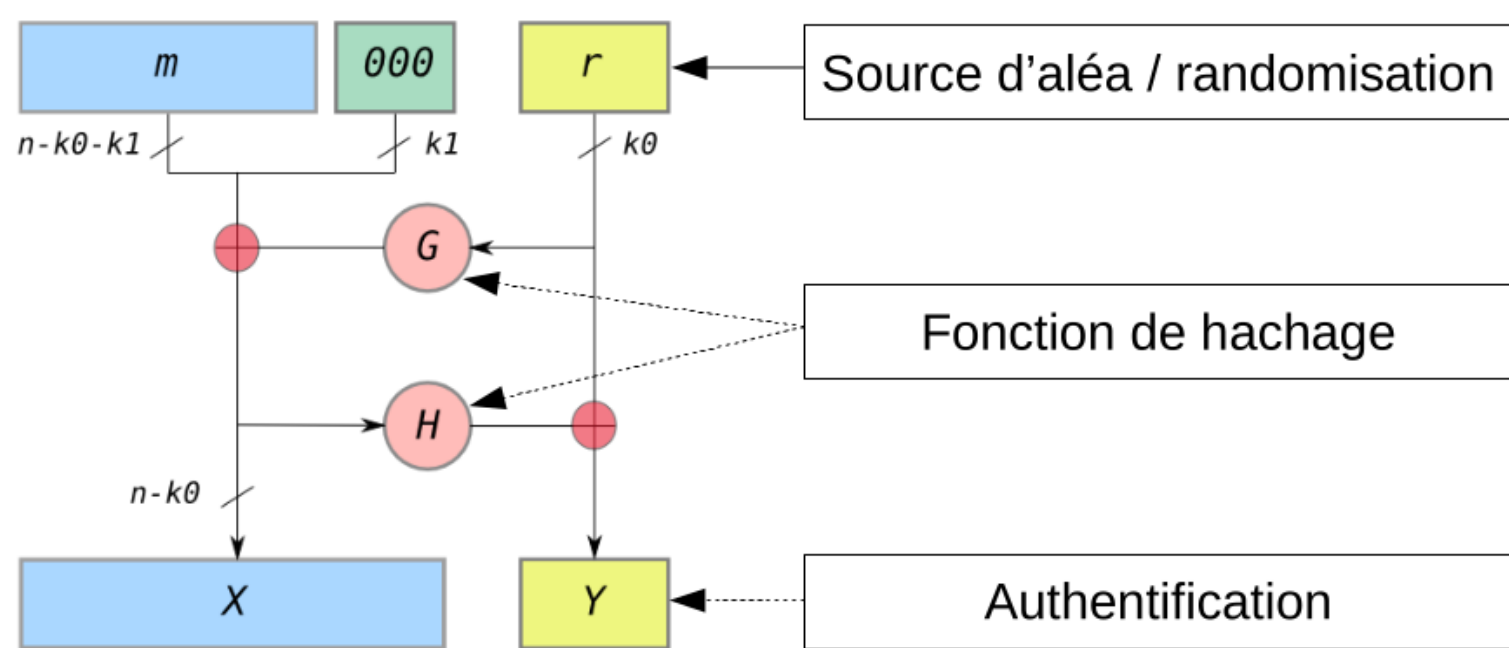
→ il faut rajouter de l'authentification → contrôle d'intégrité
→ RSA n'est pas homomorphe à la multiplication

→ rajouter de l'aléa pour que RSA ne soit plus déterministe

→ OAEP (Optimal Asymmetric Encryption Padding)

→ il faudra corriger l'aléa (XOR)

} authentification



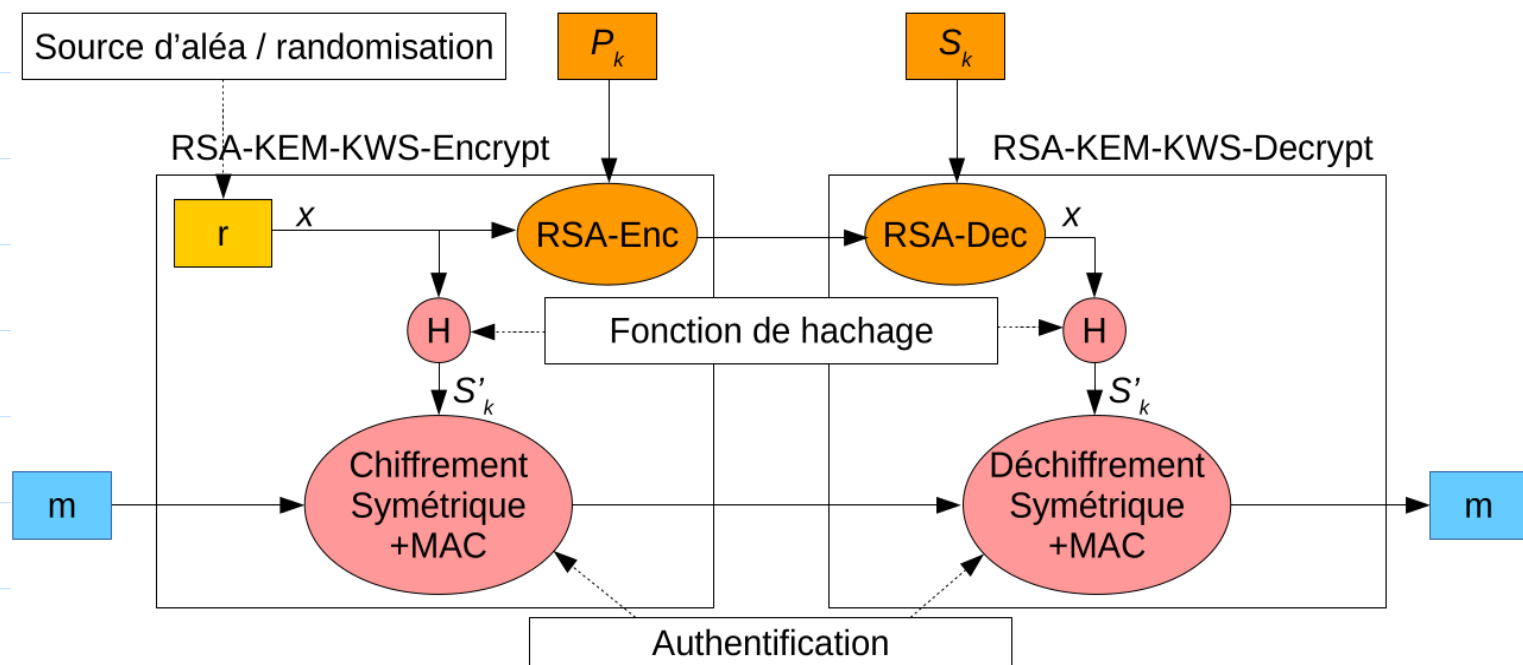
→ pourquoi on a pas gardé ça ?

→ excellente question franchement ; ai pas écarté et je sais plus

Méthode standardisée 2 (NIST SP 800 56B) :

(mécanisme à portée plus large que RSA)

Mécanisme d'encapsulation



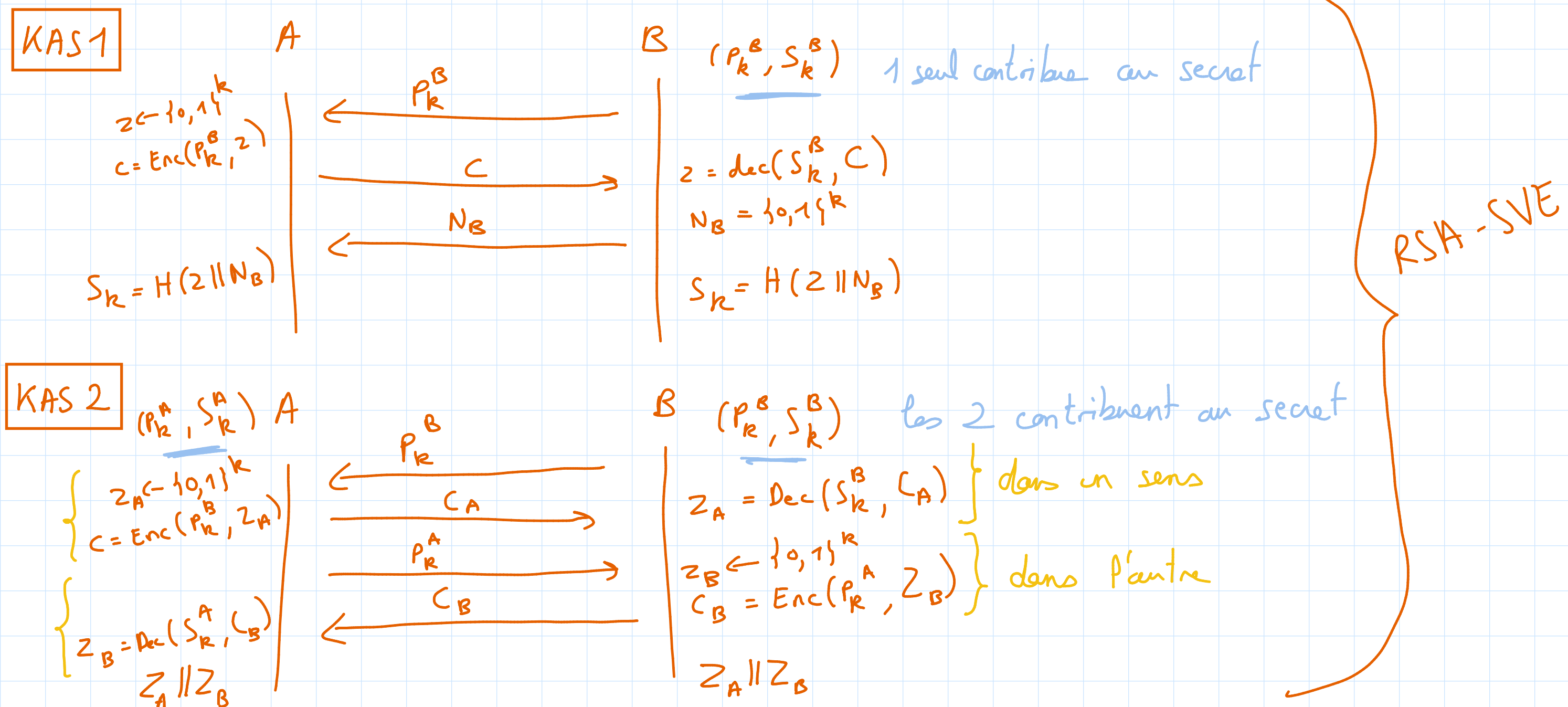
- Attques :
- sur chiffré seul : l'attaquant n'a que des chiffrés
 - clair connu (known plaintext attack) : l'attaquant a des couples (clair, chiffré)
 - clair choisi (chosen plaintext attack) : l'attaquant peut chiffrer
 - chiffré choisi (chosen ciphertext attack) : l'attaquant peut déchiffrer → si l'attaquant a les capacités de CCA, les clairs sont compromis
- IND-CCA1 : quand l'adversaire reçoit le challenge, il ne peut plus demander de déchiffrement.
 - IND-CCA2 : peut toujours demander des déchiffrements (sauf le déchiffrement du challenge tel)

→ les méthodes de randomisation passent la sécurité de RSA de IND-CPA à IND-CCA

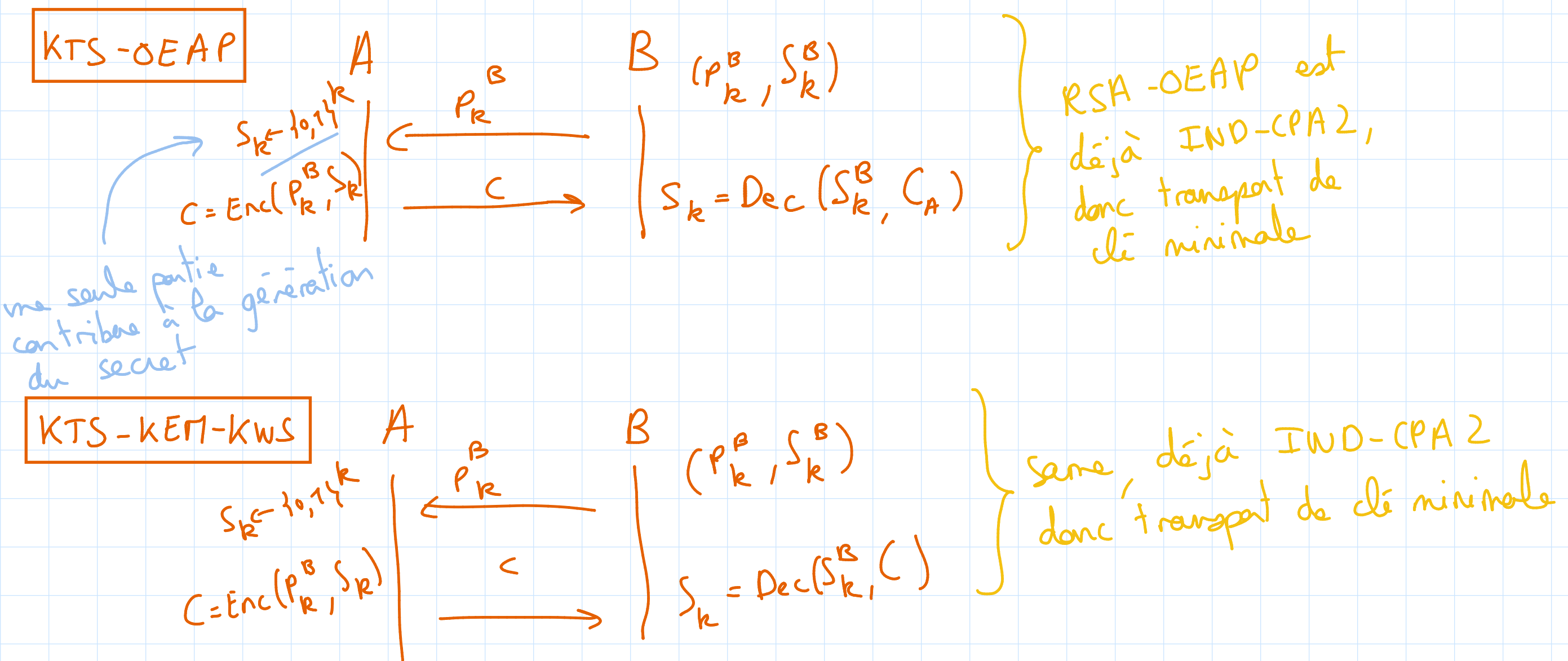
standard NIST :

- schéma accord de clé : RSA-SVE : $x \leftarrow \{0,1\}^n$, $\text{Enc}(P_k, x)$ *
- schéma transport de clé : RSA-KEM-KWS, RSA-OAEP *

* KAS1, KAS2 (accord de clé)

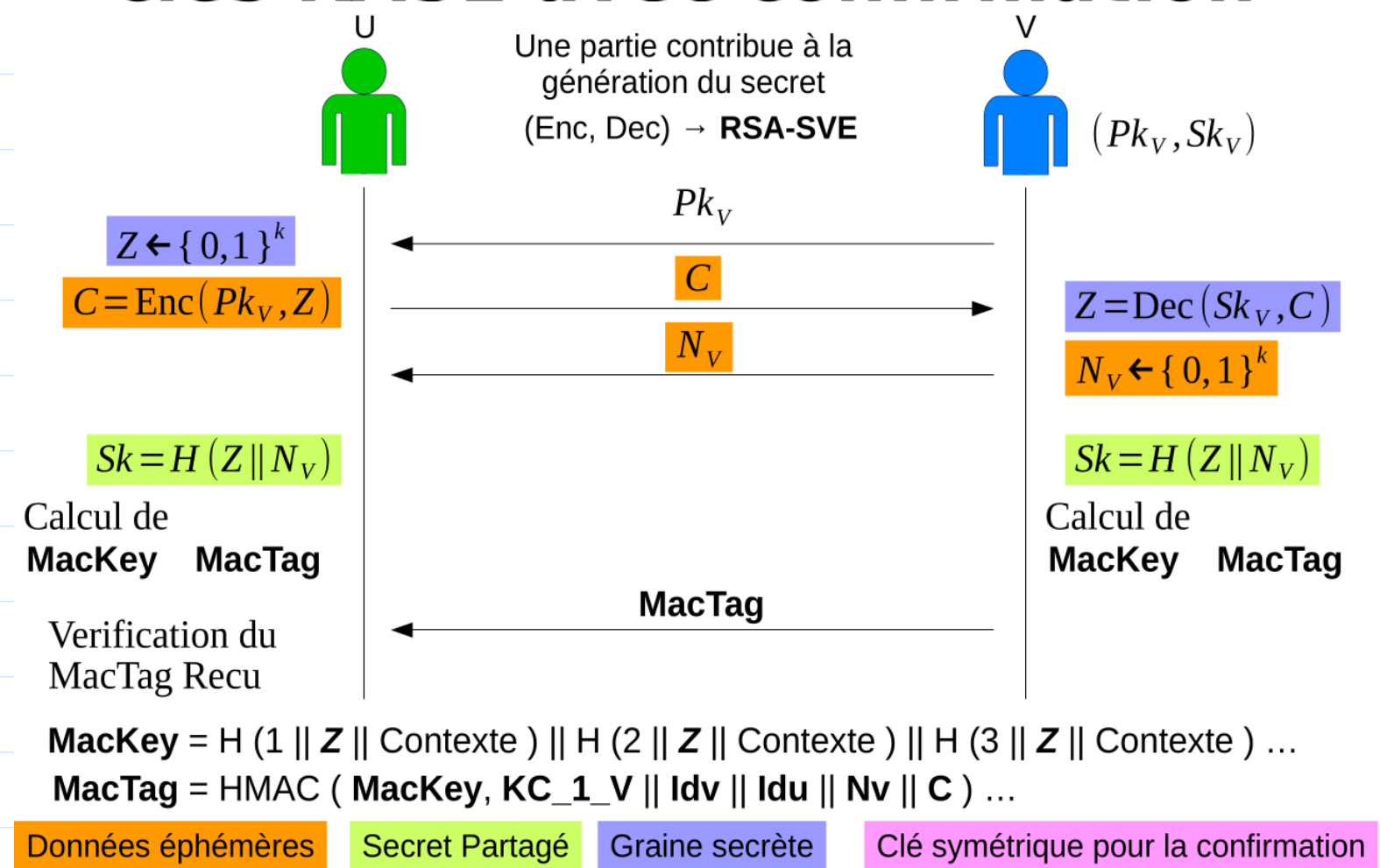


* KTS-OAEP, KTS-KEM-KWS (transport de clé)



Confirmation de clé → une sorte de hash : un truc qui permet d'identifier la clé sans rien perdre sans le secret

Exemple d'accords de clés KAS1 avec confirmation



Les algorithmes de dérivation de clé (KDF) en une étape

KDF1/MGF1 (ISO 18033-2, IEEE P1363A, PKCS#1 v2)

$CléDérivée = H(Z || 0 || Contexte) || H(Z || 1 || Contexte) || H(Z || 2 || Contexte) || \dots$
 $= H(Z || C || Contexte) || \dots$, C compteur commençant à 0.

KDF2 (ISO 18033-2, ANSI X9.42)

$CléDérivée = H(Z || C || Contexte) || \dots$, C compteur commençant à 1.

KDF3 (ISO 18033-2, NIST SP 800 56A/B)

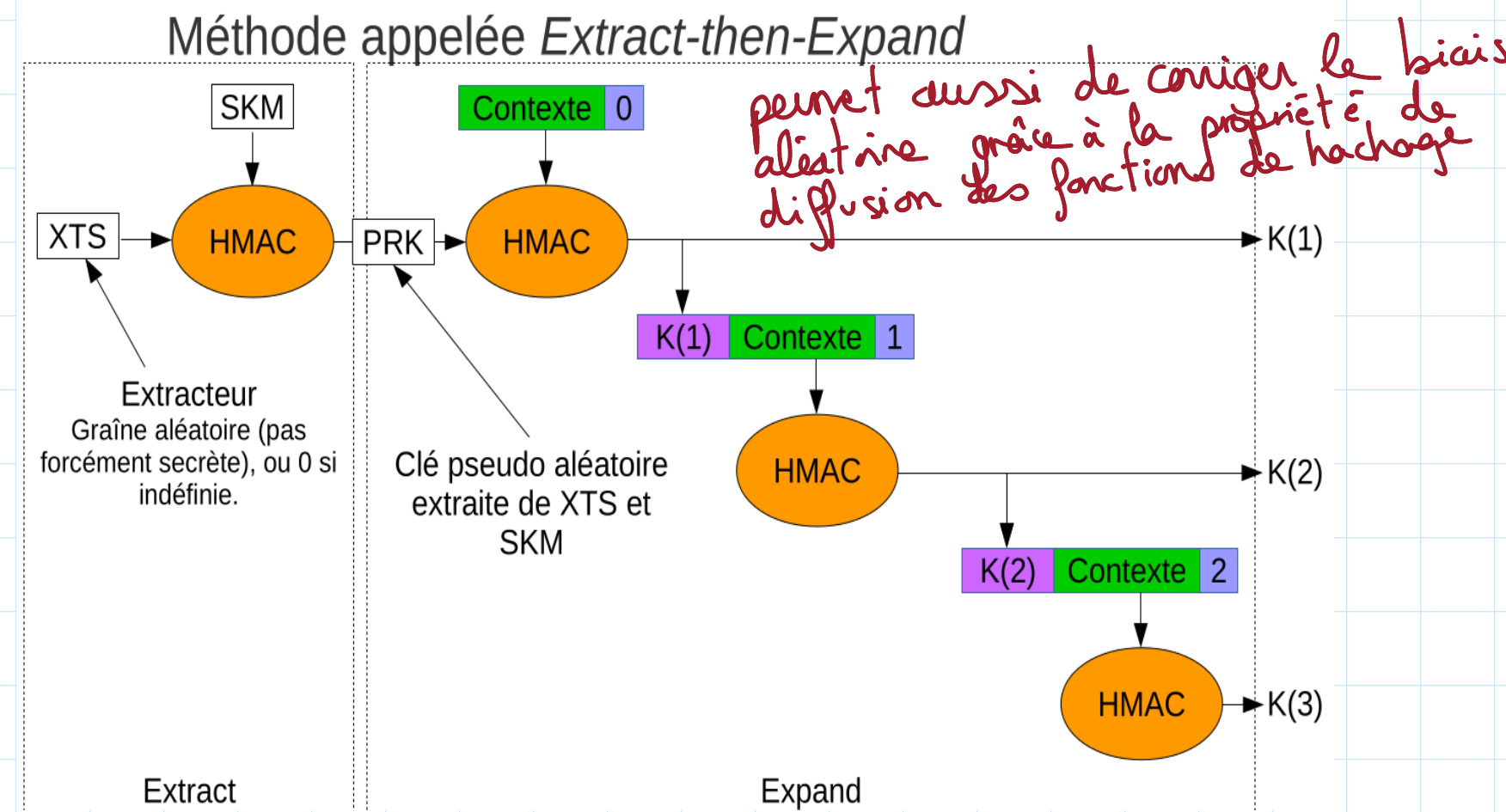
$CléDérivée = H(C || Z || Contexte) || \dots$, C compteur commençant à 0.

KDF3 (alternative NIST SP 800 56A/B)

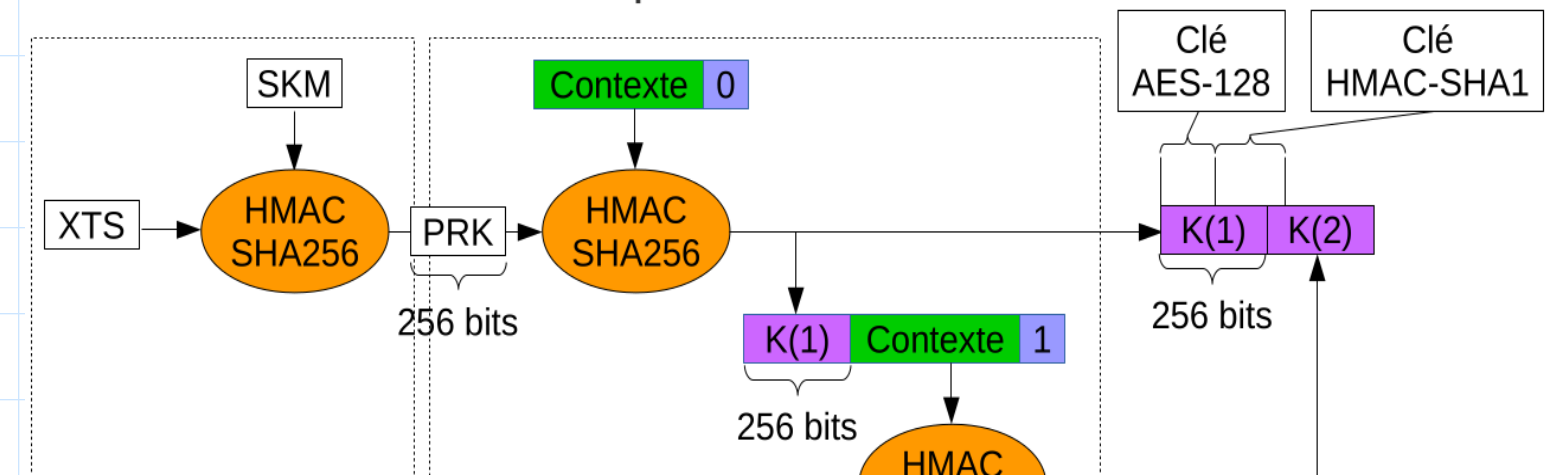
$CléDérivée = \text{HMAC}(\text{salt}, C || Z || Contexte) || \dots$, C compteur commençant à 0, salt = 0 si non défini autre part.

Les algorithmes de dérivation de clé (KDF) en deux étapes

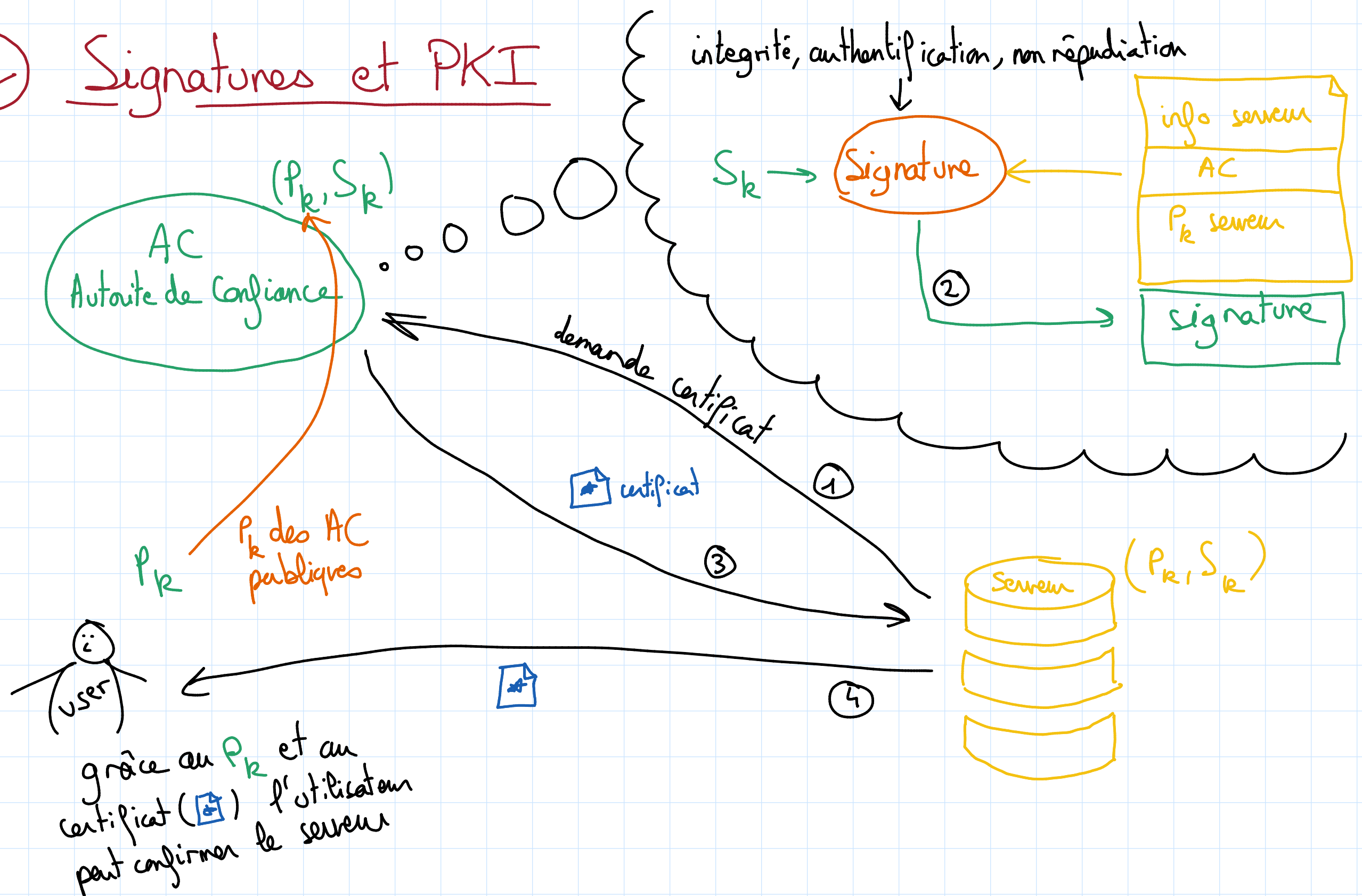
Méthode appelée *Extract-then-Expand*



Cas pratique : On utilise SHA-256 pour le HMAC afin de générer une clé pour un chiffrement AES-128 sur 128 bits, et une clé pour un HMAC-SHA1 sur 160 bits.



⑦ Signatures et PKI



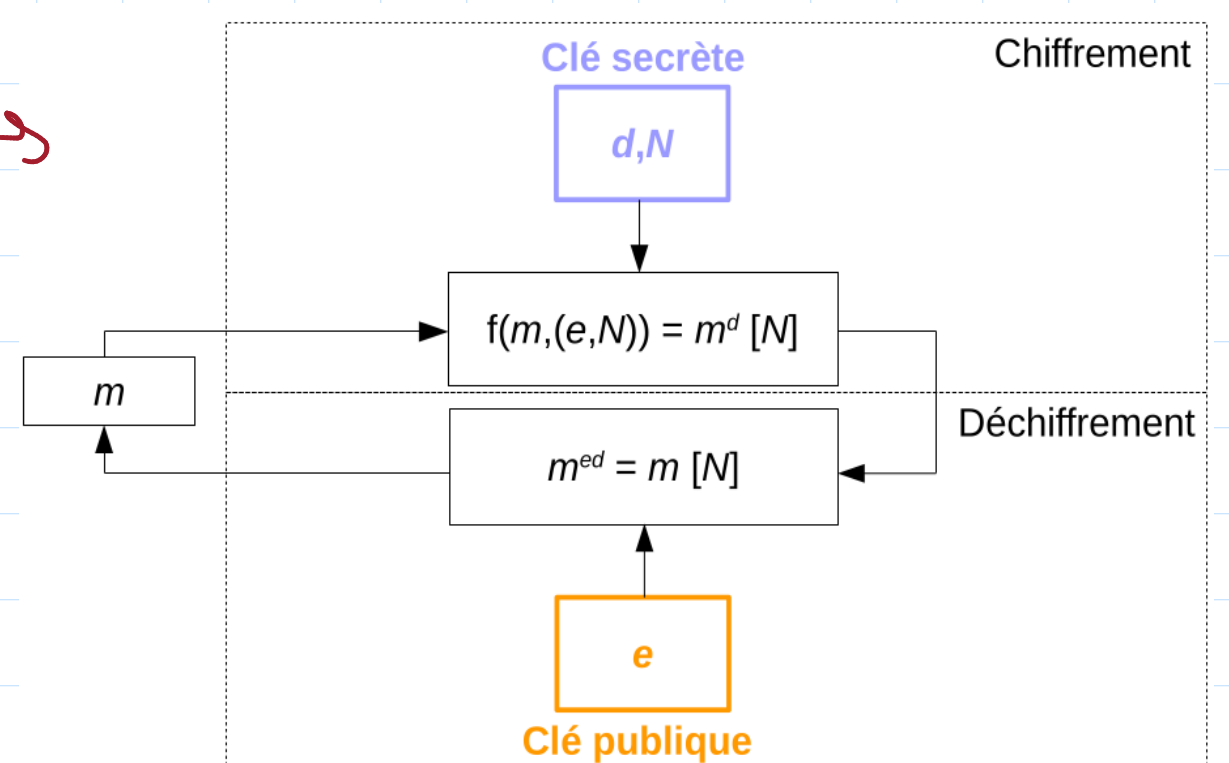
Cible en terme de sécurité	Hash	MAC	Signature
Intégrité message non altéré	✓	✓	✓
Authentification	✗	✓	✓
Non-répudiation ne peux pas dire qu'il n'est pas l'auteur du message	✗	✗	✓
Chiffrement	Aucun	Symétrique	Asymétrique

signature RSA - textbook?

NOW!

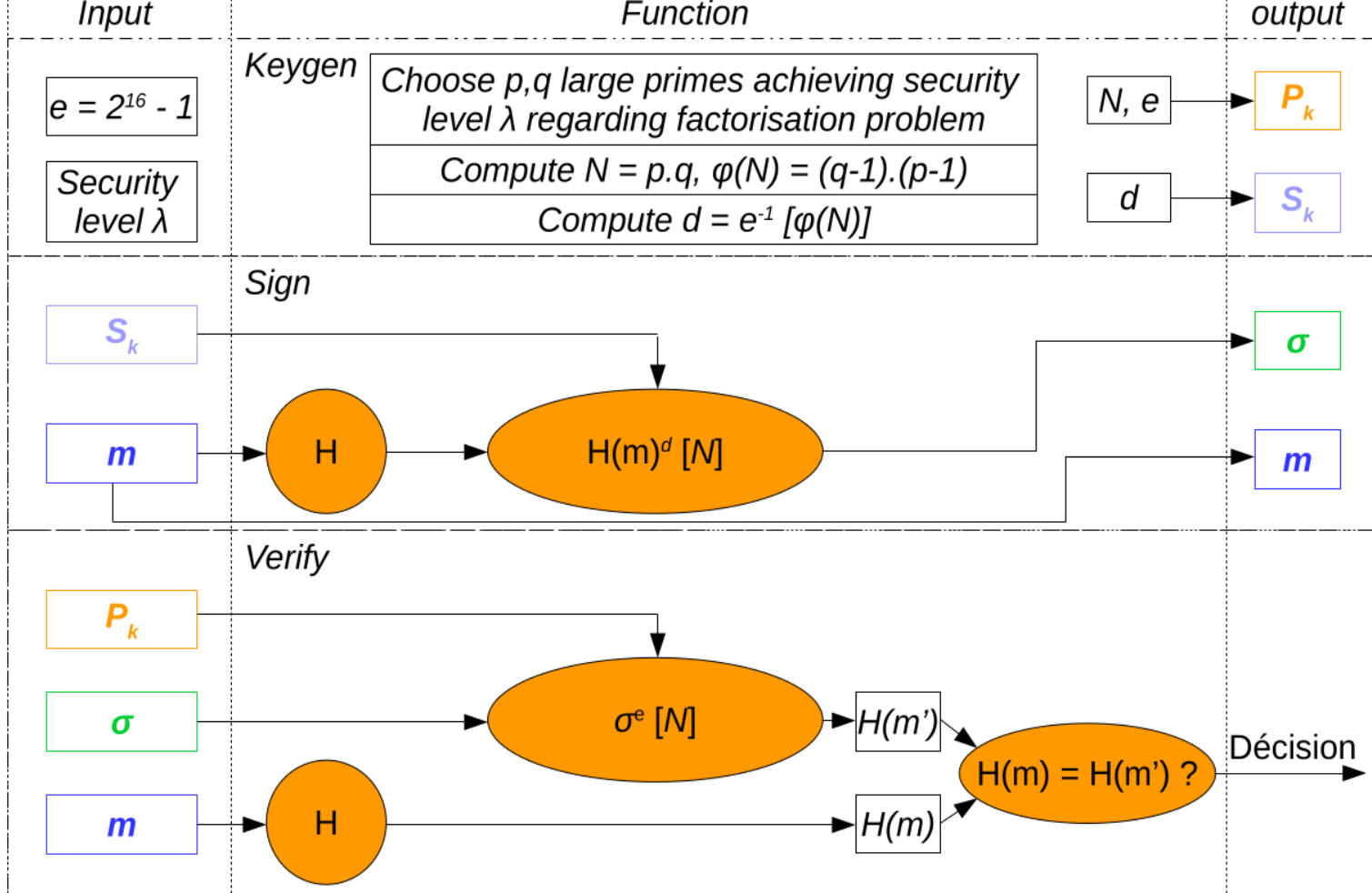
chiffrement déterministe
↓
on peut créer une signature valide

de $\oplus$: RSA homomorphe : $c_1 \cdot c_2 = (m_1 \cdot m_2)^d [N]$



RSA-FDH

← LA SOLUTION YOUPI



↳ on ne peut pas générer de signature valide sans le secret

↳ nécessite une fonction de hachage qui produise un condensat de taille modulo N de RSA

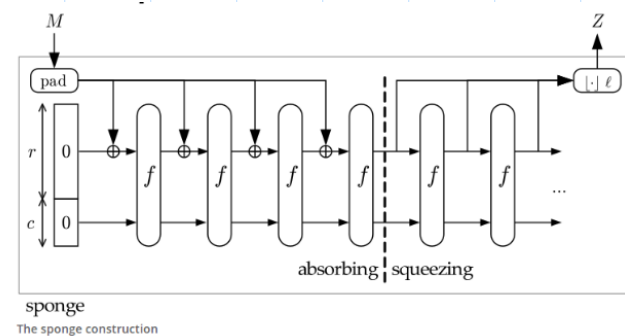
↳ sha3 par exemple

KECCAK ~ sha3

↳ absorbing and squeezing

on absorbe le message dans "l'éponge"

on essore "l'éponge" pour avoir le condensat



f : Permutation
c : Partie de l'état interne secret
r : Partie de l'état interne consacré à l'absorption/essorage

Images : site officiel Keccak

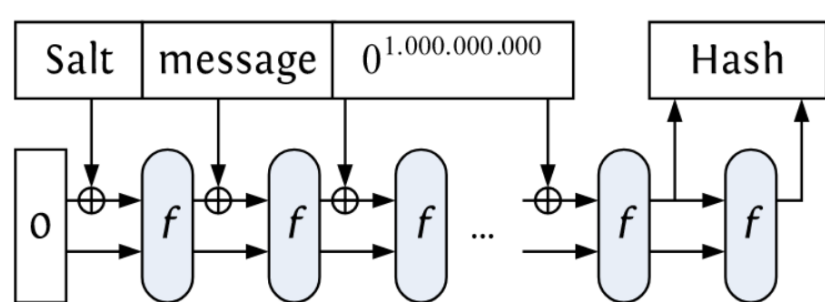
→ permet de contrôler la taille de la sortie

→ on peut rajouter du sel et/ou du padding

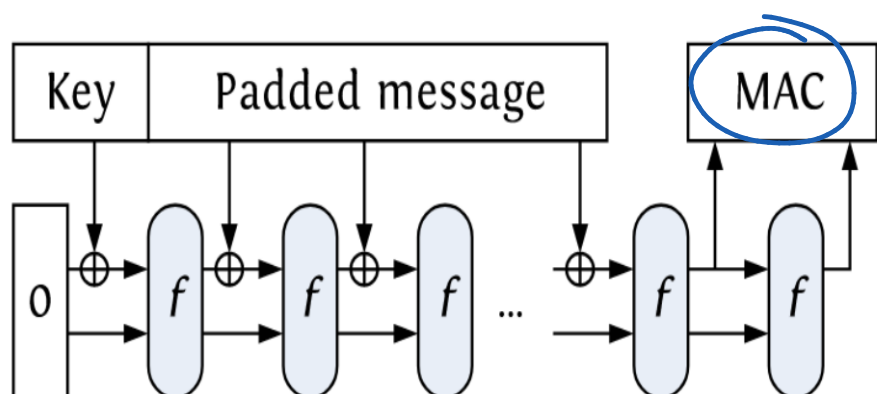


permet le temps d'exécution

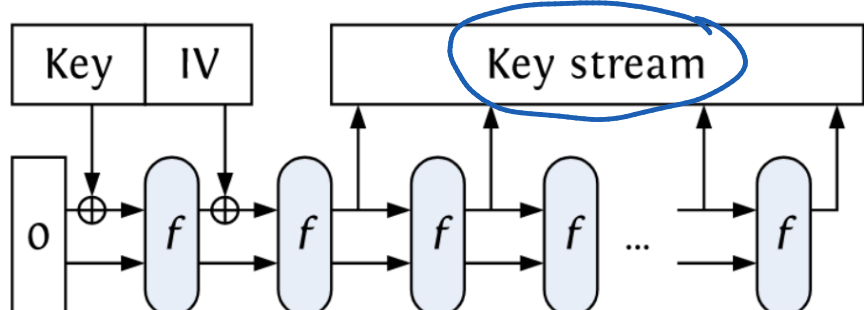
- Utilisation en fonction de hachage avec sel et pénalisation



- Utilisation en code d'authentification de message



- Utilisation en chiffrement par flot



Vérification de patches logiciels :

utilisateur

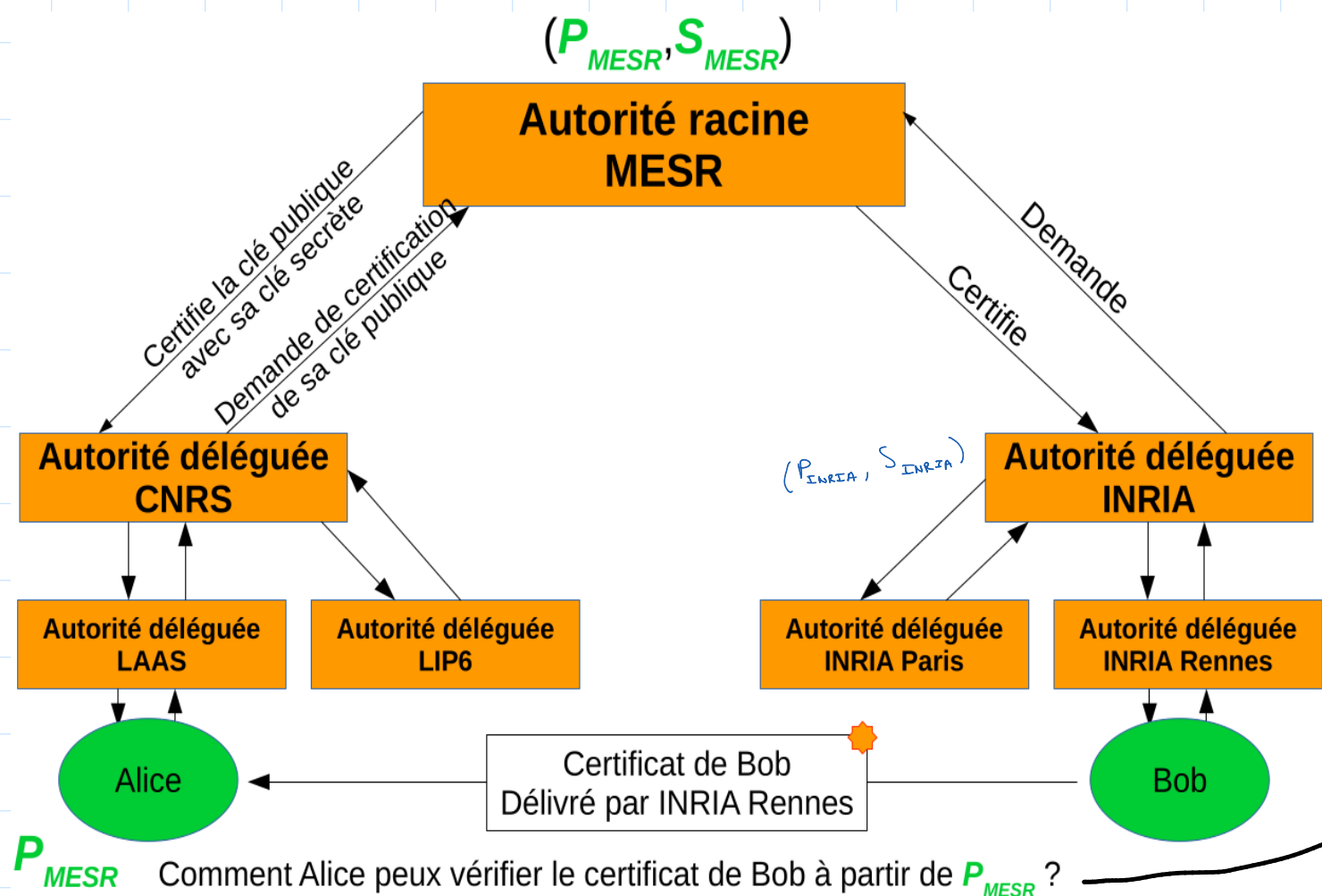
logiciel



fournisseur de logiciel
(P_k, S_k)



hiérarchie des certifications (des AC entre eux)



① on récupère le cert_{INRIA} qu'on vérifie avec P_{MESR}

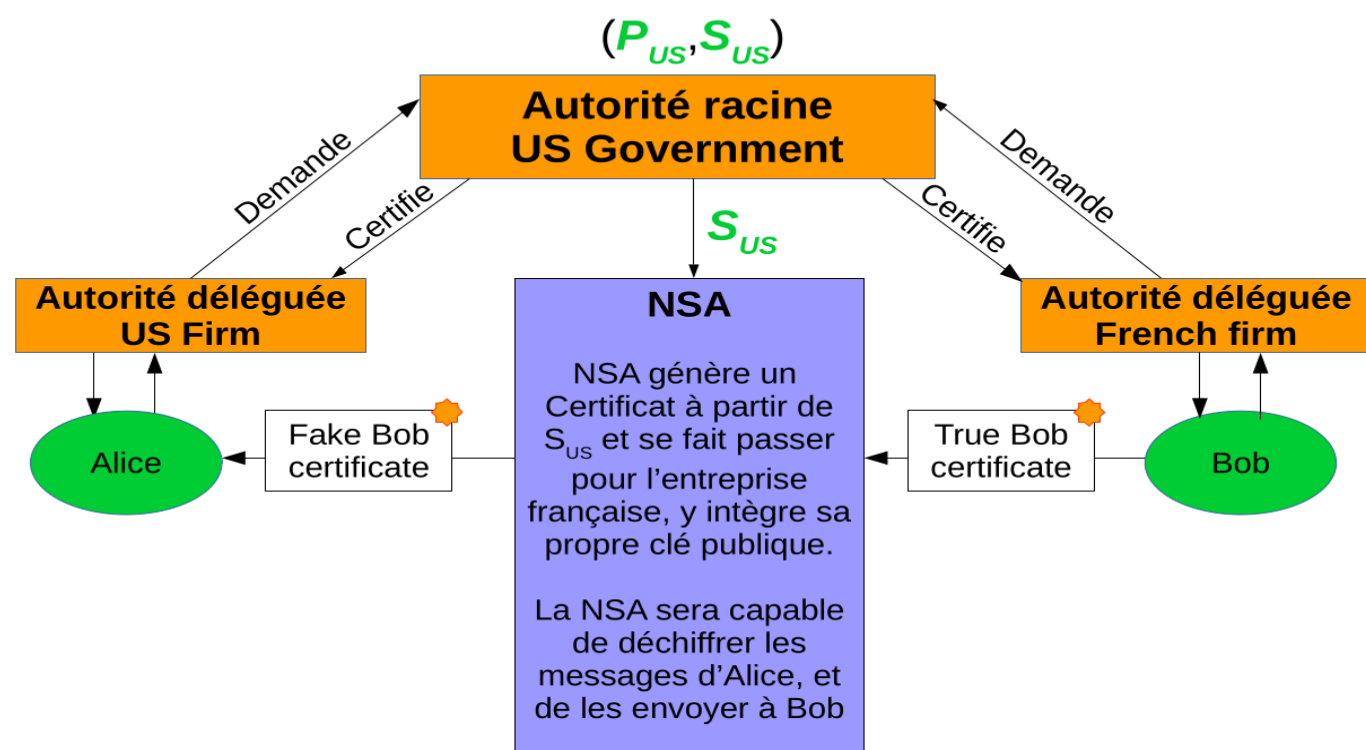
② on check avec P_{INRIA} la clé publique de INRIA Rennes

avantage : si INRIA se fait voler sa clé secrète, alors Rennes et Paris sont compromis, mais l'autorité racine (MESR) non

→ avantage du modèle hiérarchique

→ uniquement les autorités déléguées tombent

Attaque Man In The Middle



→ cas de figure :

- Proxy
- vol de clé
- "accord" secret

pour éviter ça :

- réduire confiance aux AC
- plusieurs certificats de plusieurs AC
- pinning : on n'accepte que certains certificats dit sûr (liste dans le navigateur)

Web of trust ~ principe de la toile de confiance :

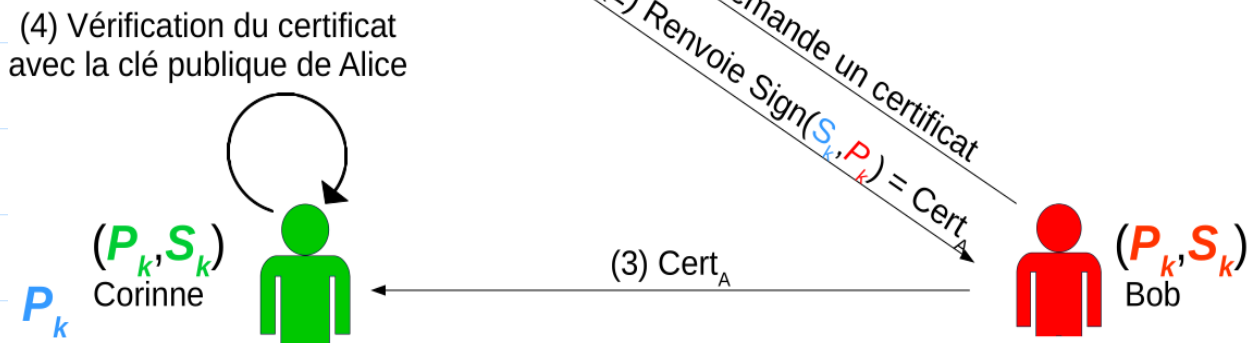
- Chaque participant agit comme un utilisateur et une autorité de confiance.
- Supposons de Corinne veuille communiquer avec Bob, sachant quelle fait confiance à Alice.
- Bob va présenter à Corinne un certificat qui a été signé par Alice. Corinne a donc confiance en ce certificat si elle peut le vérifier avec la clé publique d'Alice.

→ utilisé dans openPGP (échange de mail)

→ il faut attendre qu'un nombre suffisant de personnes nous passe confiance avant de devenir un nœud de la toile



(4) Vérification du certificat avec la clé publique de Alice



Le standard FIPS 186-4 : Digital Signature Standard

3 approches standardisées :

- **RSA :**
 - Problème de factorisation sur les corps finis.
 - Pour 112 bits de sécurité $\rightarrow \log_2 N = 2048$.
 - Utilisation proche de FDH, ou padding potentiellement randomisé.
- **DSA (Digital Signature Algorithm) :**
 - Basé sur le logarithme discret sur les corps finis
 - Pour 112 bits de sécurité $\rightarrow \log_2 p = 2048$
 - Construction de type Fiat-Shamir (hors programme).
- **ECDSA (Elliptic-Curve Digital Signature Algorithm) :**
 - Basé sur le logarithme discret sur les courbes elliptiques.
 - Pour 112 bits de sécurité $\rightarrow \log_2 p = 224$
 - Construction Fiat-Shamir également.

Principe de la révocation de certificat

Pour de bonnes raisons, il est nécessaire d'inclure un mécanisme de révocation de certificat. Exemples :

- Signifier qu'une **clé a été volée** (usurpation d'identité) ;
- Signifier qu'une **clé n'est plus valide** ;
- Inciter à la mise à jour des clés.

On utilise une liste de révocation de certificats (CRL) pour le vérifier :

- Soit on dispose d'une **liste non périmée** donc OK ;
- Soit on va **chercher la dernière CRL** ;
- Soit on utilise **un OCSP** (Online Certificate Status Protocol)

VII Side channels attacks

* modern challenges: power (P) et micro-archi c'est plus de la recherche que l'on a, moins d'outils réels opérationnels comme c'est le cas pour les couches au dessus.

* side channel attacks :

- au moment de la communication un change de bit peut produire une consommation énergétique
- $\rightarrow$ liaison entre le matériel et la consommation énergétique que l'on peut exploiter.

$\rightarrow$ il y a des archis où le noyau n'a pas accès au code (encapsulation sécurisée)

$\rightarrow$ malheureusement ce mécanisme est attaquant avec consommation énergétique

est-ce possible ? - faire un fault à répét pour faire répéter l'encodage

- faire un saut d'instruction

* canal auxiliaire : - savoir si une donnée est stockée dans le cache ou pas peut être révélateur

(ex on peut savoir si un utilisateur a visité un site ORENAS)

$\rightarrow$ on peut aussi récupérer des secrets ?

$\rightarrow$ les caches sont mutualisés au sein du processeur

* Sur les side channels attacks, l'attaquant peut savoir si on a utilisé telle ou telle adresse.

$\rightarrow$ pb critique pour AES et chiffre symétrique par des algos de pré-calcul (sensibles aux attaques de cache???)

(OpenSSL, attaque sur les serveurs)

* Note: on n'implémente pas nous-même de la crypto (il faut des connaissances transverse)

* page coloring : sécurité matérielle, chaque utilisateur a sa partie dans le cache.

* activité énergétique : R: paramètres constants + variable qui varie avec activité énergétique Nsw

* attaquant ne connaît les entrées, ni les sorties et à partir de ça peut essayer de trouver algo