Introduction à la Cryptographie

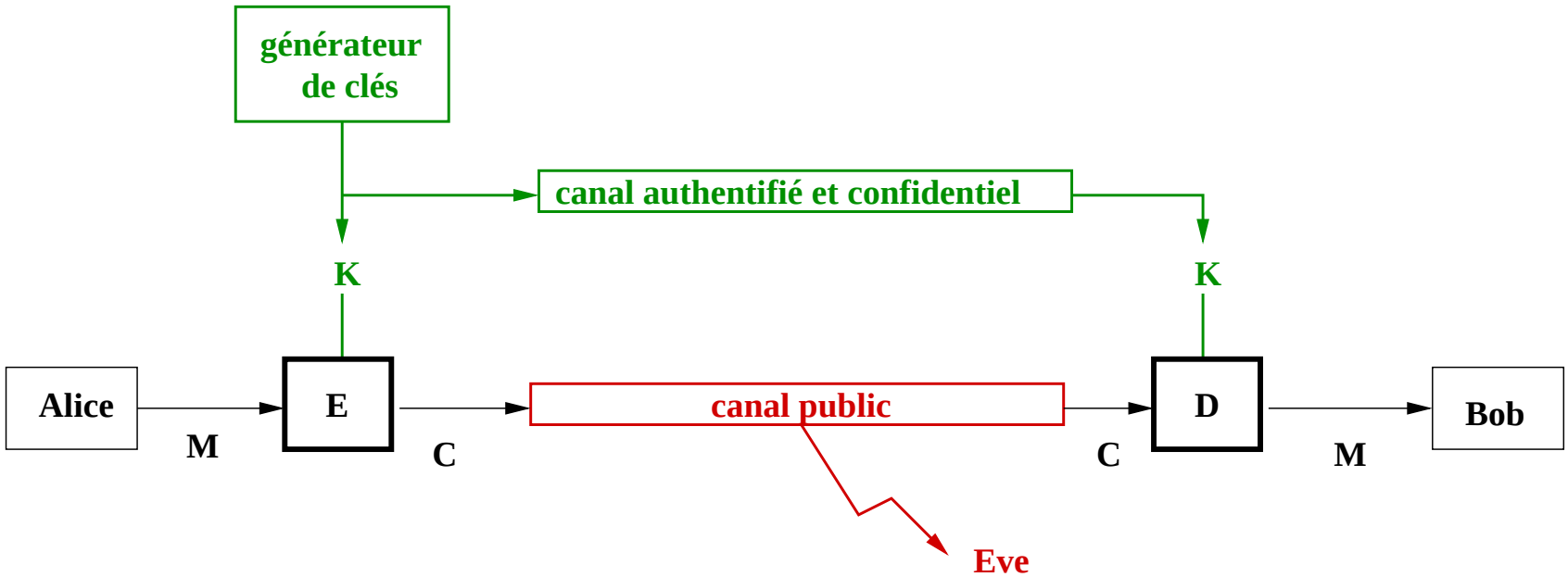
3. Cryptographie asymétrique

Cécile Pierrot, Chargée de Recherche INRIA Nancy
cecile.pierrot@inria.fr

Supports de E. Thomé

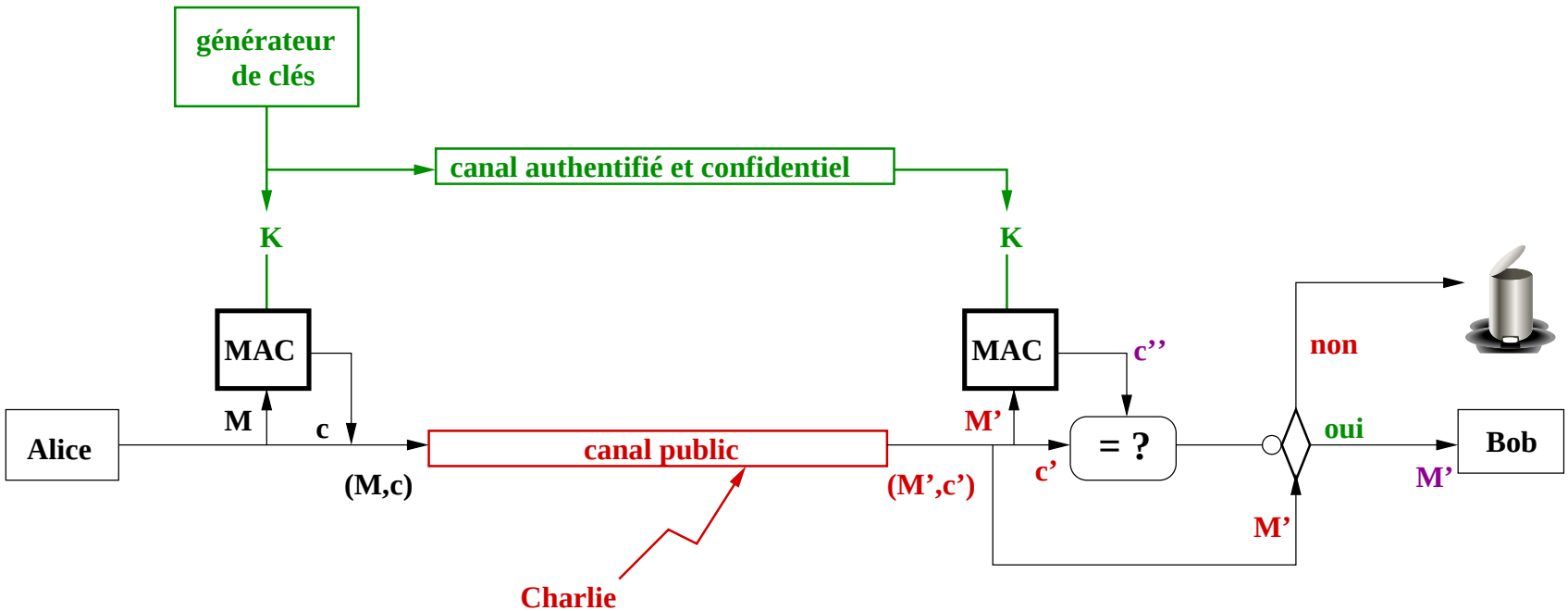
Telecom Nancy, 2A ISS – 2021

Créer un canal confidentiel



- On a besoin du canal authentifié et confidentiel au préalable de l'envoi d'un nombre élevé de messages

Créer un canal authentifié



- On utilise le canal authentifié et confidentiel préalablement au message

Observations

Deux défauts majeurs du contexte symétrique :

- Difficulté de l'échange de clés initial.
- Symétrie : si A et B connaissent K, il peuvent faire la même chose avec.
  - Pas possible de signer
  - Une clé par couple (A, B). n^2 clés pour n participants.

Plan

Clé publique

RSA et la factorisation

Diffie - Hellman et le problème du logarithme discret.

Crypto à clé publique

La cryptographie à clé publique rend possible le contexte où :

- Alice dispose d'une clé de chiffrement k_A, publique.
- Bob dispose d'une clé de déchiffrement k_B, secrète.

Cette asymétrie rend notamment possible la signature.

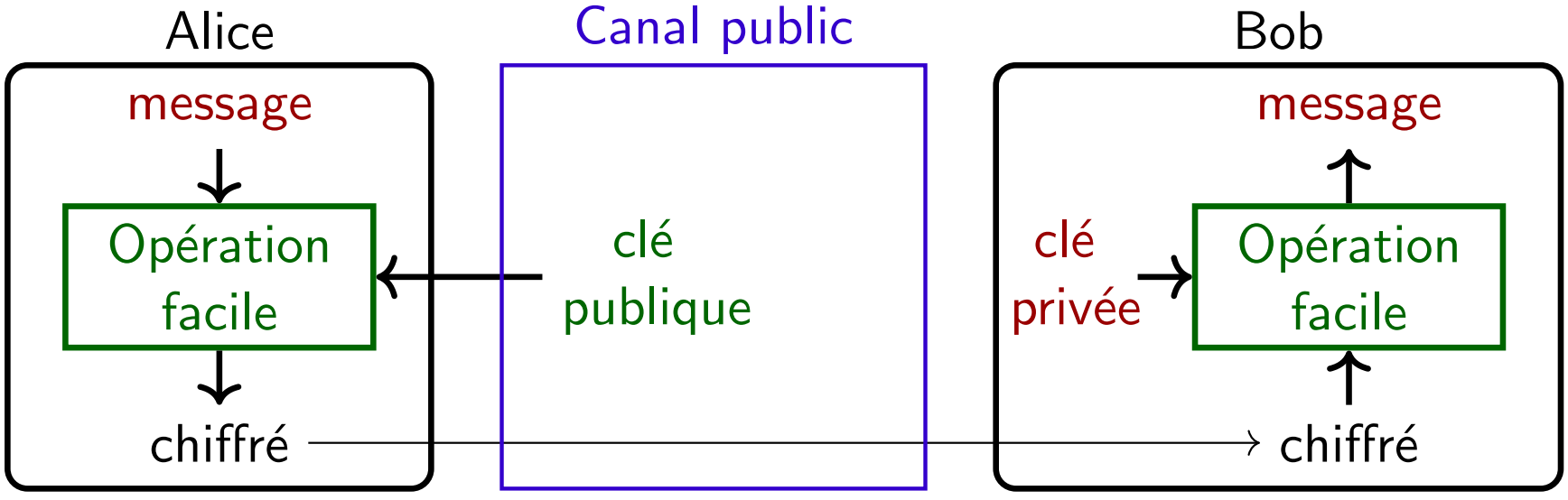
Outils

La cryptographie à clé publique repose sur la difficulté de problèmes mathématiques.

- En connaissant k_A, il est possible de chiffrer.
- En connaissant k_B, il est possible de déchiffrer.
- Le problème difficile est : découvrir k_B à partir de k_A :
  - calculatoirement hors de portée
  - (mais possible en un temps infini).

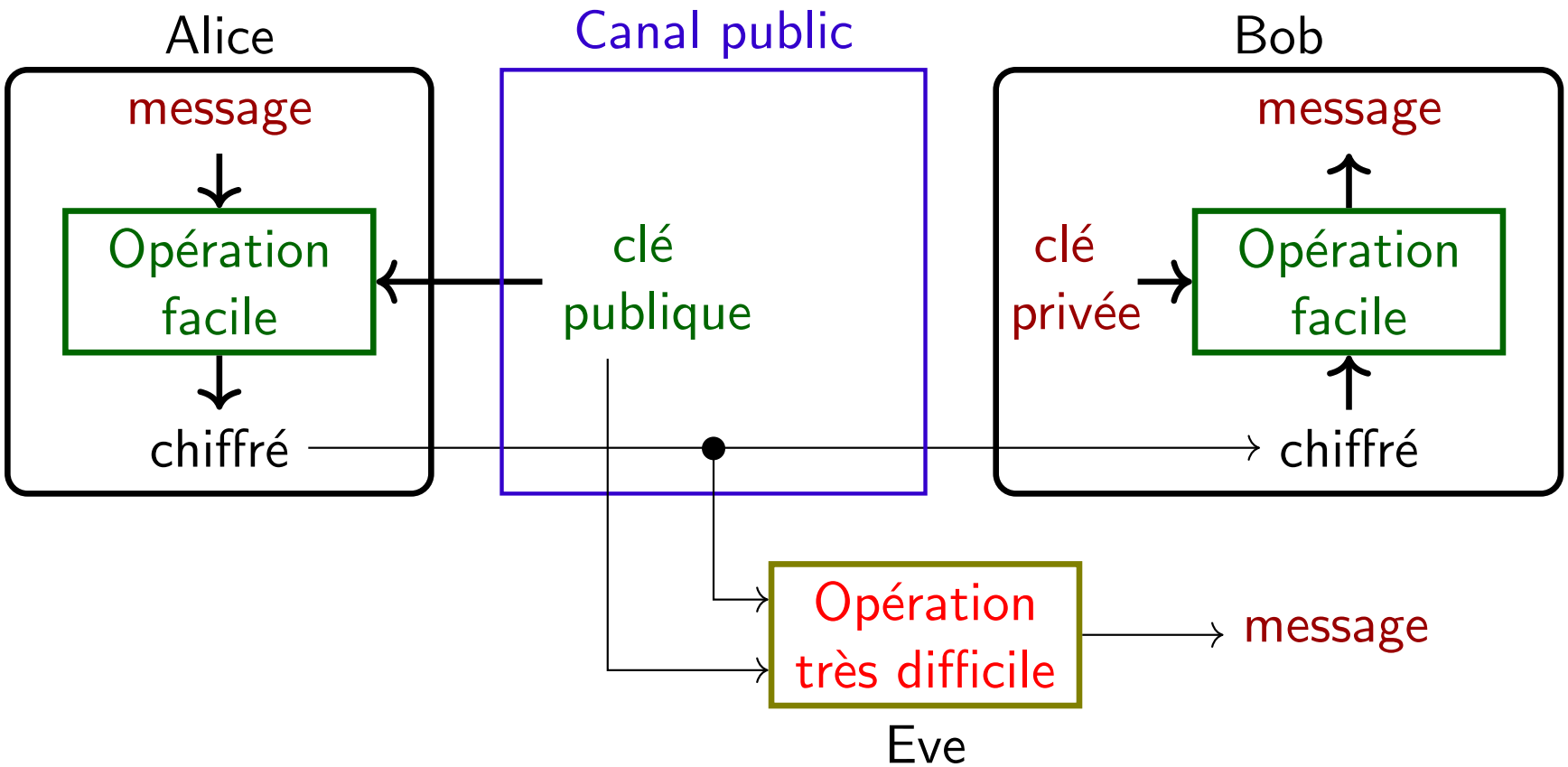
Reposer sur un problème mathématique (1)

Alice envoie un message chiffré à Bob.



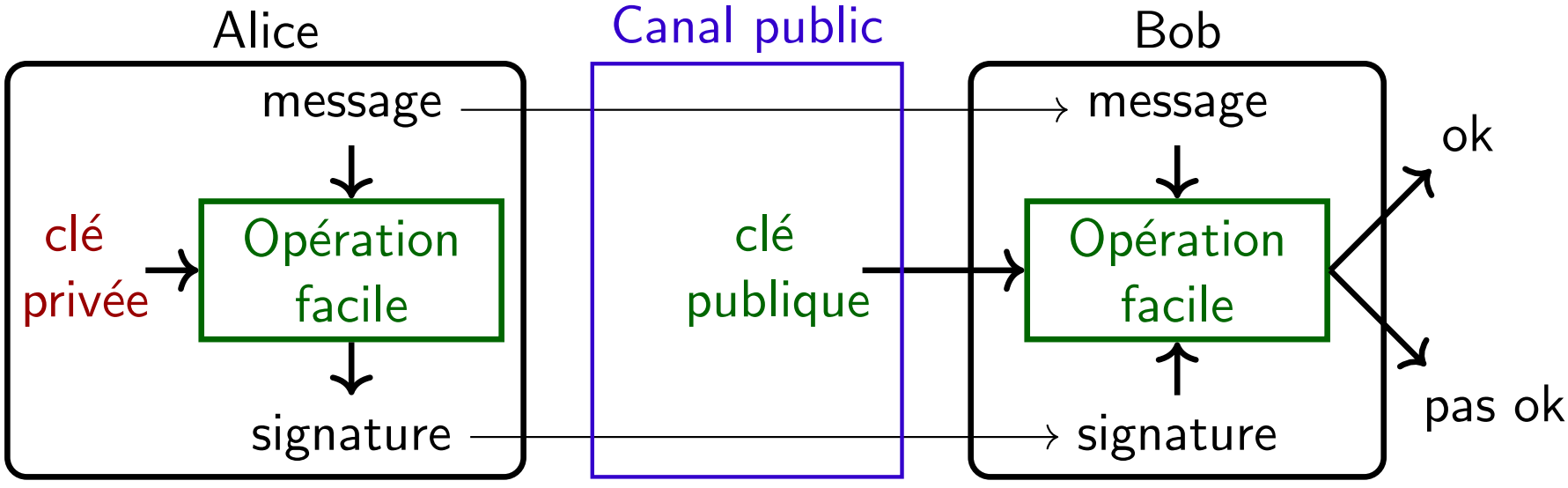
Reposer sur un problème mathématique (1)

Alice envoie un message chiffré à Bob.



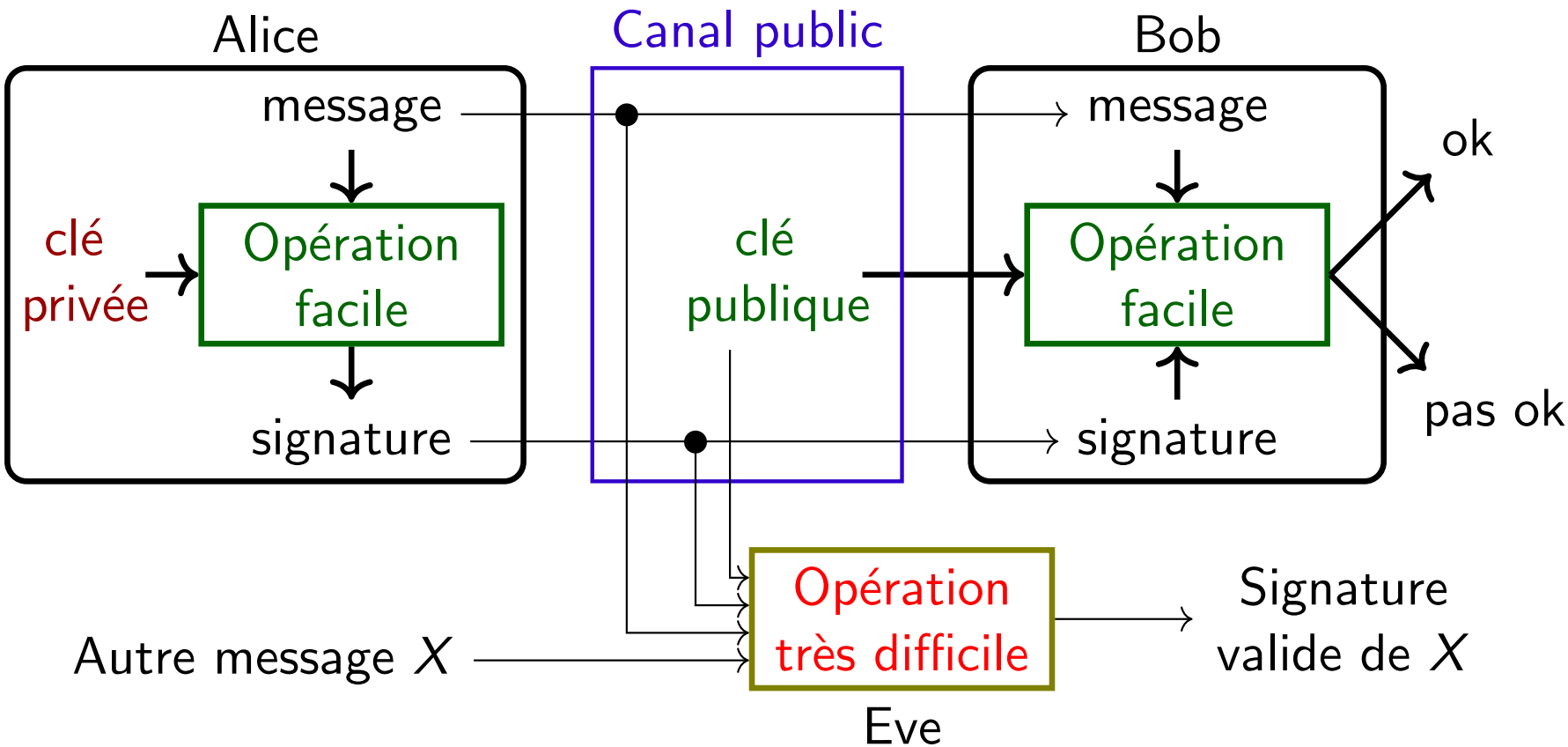
Reposer sur un problème mathématique (2)

Bob vérifie un message signé par Alice.



Reposer sur un problème mathématique (2)

Bob vérifie un message signé par Alice.



Outils

Les problèmes mathématiques couramment rencontrés en crypto.

Pour les plus fréquents :

- Problème de la factorisation d'entiers (cryptosystème RSA).
- Problème du calcul de logarithme discret dans les groups (échange de clés).

Mais aussi, en post quantique :

- Problèmes liés à la théorie des codes correcteurs.
- Recherche de vecteurs courts dans des réseaux de $\mathbb{Z}^n$.
- Problèmes sur les Isogénies, les fonctions de hachage etc

Systèmes étudiés

On présente :

- Le cryptosystème RSA : Rivest, Shamir, Adleman
- Le protocole de Diffie-Hellman.

Plan

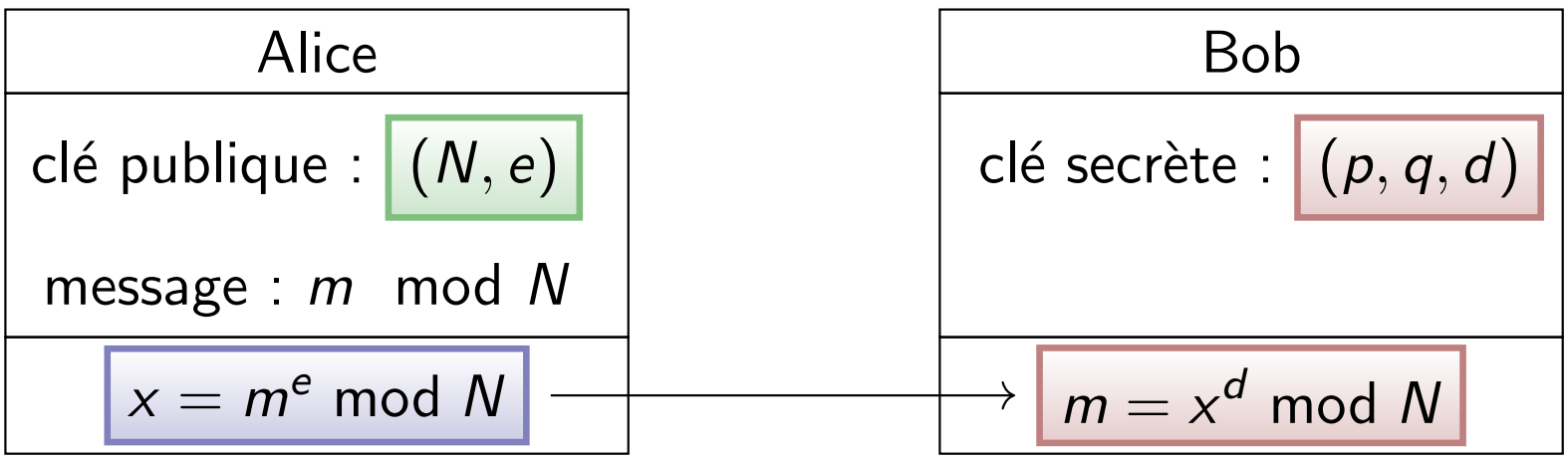
Clé publique

RSA et la factorisation

Diffie - Hellman et le problème du logarithme discret.

Le chiffrement RSA

- N entier, et p, q (premiers) tels que $N = pq$.
- $\phi(N) = (p-1)(q-1)$
- e premier avec $\phi(N)$. L'entier e est inversible modulo $\phi(N)$.
- d calculé tel que $de \equiv 1 \mod \phi(N)$.



$$x^d \equiv m^{ed} \equiv m^{1+k\phi(N)} \equiv m \mod N.$$

RSA : exemple

En TD !

Signature RSA

Pour signer un message m :

- Bob calcule $s = m^d \mod N$.
- Alice vérifie que $s^e \equiv m \mod N$.

Preuve de RSA

On a $ed \equiv 1 \mod \phi(N)$, et $\phi(N) = (p-1)(q-1)$.

En particulier, on a $ed = 1 + k(p-1)(q-1)$, donc $m^{ed} = m^{1+(p-1)\times \text{cst.}}$.

- Petit théorème de Fermat : $m^{p-1} \equiv 1 \mod p$.
  Donc $m^{ed} \equiv m \mod p$.
- Si jamais $p \mid m$, alors $p \mid m^{ed}$, donc $m^{ed} \equiv m \mod p$.

Idem pour q, donc $pq \mid (m^{ed} - m)$, $\square$.

Que peut faire un espion ?

- S'il sait factoriser N, il a gagné.
- S'il ne peut pas, est-il impossible de calculer $\sqrt[e]{x \bmod N}$?
- Factoriser N est la meilleure approche connue pour $\sqrt[e]{}$.
- Miracle RSA : On ne sait pas factoriser N lorsqu'il devient grand.

Le choix des clés pour RSA

- Un module RSA est un nombre composé N.
- Un module RSA ne sert qu'à une seule personne.

Il faut définir une procédure pour générer des clés.

- Choisir un nombre premier p aléatoire de $n/2$ bits.
- Choisir un nombre premier q aléatoire de $n/2$ bits.
- Choisir e, calculer d tel que $de \equiv 1 \mod (p-1)(q-1)$.
- Publier N, e, conserver p, q, d.

Le choix du nombre de bits n est crucial.

Choix des clés

Factoriser N est difficile lorsqu'il devient assez grand. Grand comment ?

- Si on veut de la sécurité dans la durée, il faut avoir une idée de la difficulté de la factorisation dans le futur.
- Record de factorization : N de 829 bits, en 2020. Environ 2700 ans CPU ! L'algorithme d'attaque utilisé est le crible algébrique (NFS) d'une complexité exp $\left(O(\sqrt[3]{n})\right)$.
- Aujourd'hui : l'ANSSI ne valide pas l'usage de clés RSA de moins de 2048 bits dans les produits crypto (3072 si usage après 2030).
- Il existe encore des clefs de 1024 bits... correspond à 2^{77} opérations, à éviter !

Plan

Clé publique

RSA et la factorisation

Diffie - Hellman et le problème du logarithme discret.

S'appuyer sur un problème mathématique

RSA : on s'appuie sur un problème mathématique : la factorisation.

Un problème cousin : le problème du logarithme discret.

Exemple dans un groupe cyclique $G = 1, g, g^2, g^3, \cdots, g^{n-1}$.

- Une opération facile : la puissance.
- Une opération difficile : le logarithme discret.

Logarithme discret

Le logarithme discret de $a \in G$ est l'élément $k \in \mathbb{Z}/n\mathbb{Z}$ tel que :

$$g^k = a.$$

C'est l'opération inverse de l'exponentiation.

En général, c'est très difficile.

À quoi cela est-il utile ?

L'asymétrie entre exponentiation et logarithme discret sert à fabriquer des protocoles.

But du jeu :

- Tâche facile = exponentiation = tâche de l'acteur honnête.
- Tâche difficile = log discret = tâche de l'attaquant.

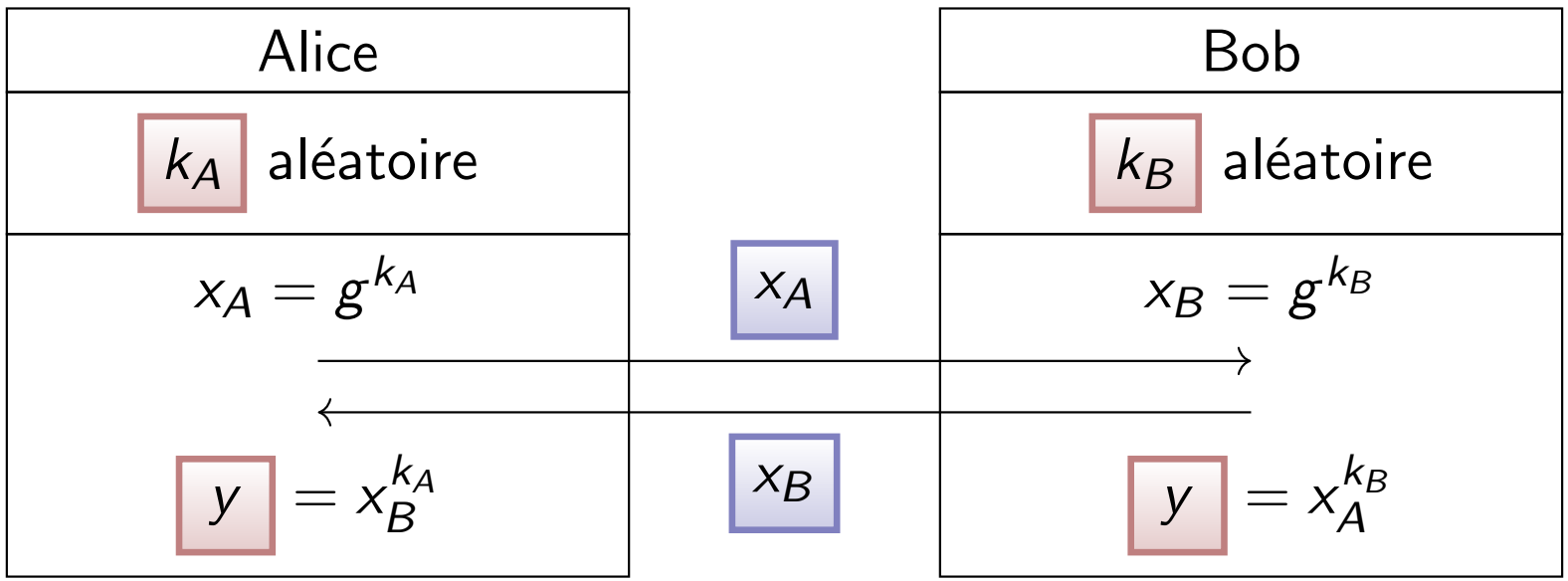
On étudie plusieurs primitives :

- Échange de clés.
- Chiffrement.
- Signature.

Diffie-Hellman

Primitive essentielle : l'échange de clés.

Donnée publique : un groupe $G = \langle g \rangle$, avec $\#G = n$.



Alice et Bob peuvent ensuite utiliser y comme clé de chiffrement pour leur communication (avec AES par exemple).

Sécurité de Diffie-Hellmann

Un espion potentiel doit retrouver $g^{k_A k_B}$ à partir de g^{k_A} et g^{k_B}.

- Résultats partiels d'équivalence avec $g^x \rightsquigarrow x$.
- $g^x \rightsquigarrow x$: problème du logarithme discret.
- Le choix d'un groupe adéquat est crucial.

Chiffrement : ElGamal

Donnée publique : un groupe $G = \langle g \rangle$, avec $\#G = n$.

Alice choisit au hasard une clé secrète $s \in \mathbb{Z}/n\mathbb{Z}$, et publie $h = g^s$.

Chiffrement : ElGamal

Donnée publique : un groupe $G = \langle g \rangle$, avec $\#G = n$.

Alice choisit $s \in_R \mathbb{Z}/n\mathbb{Z}$, et publie $h = g^s$.

Chiffrement : ElGamal

Donnée publique : un groupe $G = \langle g \rangle$, avec $\#G = n$.

Alice choisit $s \in_R \mathbb{Z}/n\mathbb{Z}$, et publie $h = g^s$.

Quand Bob souhaite envoyer un message m à Alice :

- Bob calcule au hasard $r \in \mathbb{Z}/n\mathbb{Z}$, et envoie $(g^r, h^r m)$.
- Alice recoit (u, v), et calcule $v \cdot u^{-s} = m$.

Tâche de l'attaquant : $(g^r, g^s) \rightsquigarrow h^r = g^{rs}$.

Attention !

Bob doit être sérieux dans son choix d'aléa.
Le cryptosystème d'ElGamal est très demandeur d'aléa !

Signature : Digital Signature Algorithm

Donnée publique :

- Un groupe $G = \langle g \rangle$, avec $\#G = n$.
- Une fonction arbitraire $\phi : G \to \mathbb{Z}/n\mathbb{Z}$ « pas trop moche ».
- Une fonction de hachage H.

Alice : clé secrète $s \in_R \mathbb{Z}/n\mathbb{Z}$, clé publique $h = g^s$.

Signature de m : couple $(u \in \mathbb{Z}/n\mathbb{Z}, v \in \mathbb{Z}/n\mathbb{Z})$, v premier avec n, tels que :

$$\phi(g^{H(m)v^{-1}} h^{uv^{-1}}) = u.$$

Alice la calcule avec :

- Choix de $k \in_R \mathbb{Z}/n\mathbb{Z}$,
- $u = \phi(g^k)$, et $v = k^{-1}(H(m) + su)$.

Signature DSA

Job d'Alice :

- $k \in_R \mathbb{Z}/n\mathbb{Z}$. — Tirage aléatoire.
- g^k. — Une exponentiation dans G.
- $u = \phi(g^k)$. — Une application de ϕ.
- $H(m)$. — Fonction de hachage.
- $H(m) + s \cdot u$. — Produit dans $\mathbb{Z}/n\mathbb{Z}$.
- $k^{-1} \mod n$. — Inversion dans $\mathbb{Z}/n\mathbb{Z}$.
- $v = k^{-1}(H(m) + su)$. — Produit dans $\mathbb{Z}/n\mathbb{Z}$.

Job de Bob (vérifier) :

- $H(m)v^{-1}$ — Hachage, inverse, produit.
- uv^{-1} — produit.
- $g^{H(m)v^{-1}}$ — Une exponentiation dans G.
- $h^{uv^{-1}}$ — Une exponentiation dans G.
- $\phi(g^{H(m)v^{-1}} h^{uv^{-1}})$ — Un produit dans G, et une comparaison.

DSA = une horreur ?

C'est trop compliqué ?

- Oui : je suis incapable de me souvenir par cœur des formules.
- Pas tant : je sais où se trouve l'information, c'est cela qui compte.
- Compliqué à décrire n'est pas incompatible avec rapide !

Pour implanter DSA, il faut savoir :

- Calculer dans $\mathbb{Z}/n\mathbb{Z}$ (= modulo n).
- Calculer dans G.
- Appliquer les fonctions ϕ et H.
- Faire un tirage aléatoire.

NB : Signature = éléments de $\mathbb{Z}/n\mathbb{Z}$, pas de G. Parfois nettement plus court à écrire.

Choix de groupes

Groupes plus résistants :

- Groupes multiplicatifs de corps finis : par ex. les entiers modulo p.
- Des points sur une courbe elliptique.

Ces groupes se distinguent par la taille de groupe nécessaire pour contrer les attaques sur le logarithme discret.

Éléments de comparaison : vitesse et sécurité par rapport à RSA.

Quelles sont les meilleures méthodes (aujourd'hui) pour attaquer ?

Les algorithmes de calcul de logarithmes discret mettent en danger les protocoles.

- Un attaquant qui résout le pb du log. di. ≪ casse ≫ Diffie-Hellman.

Quelles sont les meilleures méthodes (aujourd'hui) pour attaquer ?

Les algorithmes de calcul de logarithmes discret mettent en danger les protocoles.

- Un attaquant qui résout le pb du log. di. $\ll$ casse $\gg$ Diffie-Hellman.
- Un attaquant qui résout le pb du log. di. $\ll$ casse $\gg$ ElGamal.

Quelles sont les meilleures méthodes (aujourd'hui) pour attaquer ?

Les algorithmes de calcul de logarithmes discret mettent en danger les protocoles.

- Un attaquant qui résout le pb du log. di.
  ≪ casse ≫ Diffie-Hellman.
- Un attaquant qui résout le pb du log. di. ≪ casse ≫ ElGamal.
- Un attaquant qui résout le pb du log. di. ≪ casse ≫ DSA.

Un algorithme de calcul de logarithme discret. :

- s'applique à une famille particulière de groupes.
- a une complexité qui dépend de la taille du groupe.

Certains algorithmes sont plus spécifiques que d'autres.
La bonne taille de groupe dépend de la famille.

Les algorithmes «génériques»

Les algorithmes les moins spécifiques s'appliquent à tous les groupes.

Énoncé général : $G = \langle g \rangle$ connu, $\#G = n$; $a = g^x$, trouver x.

Exemple des « pas de bébé, pas de géant » :

- Soit $C = \lceil \sqrt{n} \rceil$. x est inconnu, mais on sait qu'il existe une écriture $x = Cy + z$, avec $y, z \leq C$.
- Soit $\mathcal{B} = \{1, g, g^2, \ldots, g^C\}$, et $\mathcal{G} = \{ag^{-C}, ag^{-2C}, \ldots, ag^{-C^2}\}$.
- Il existe $\beta \in \mathcal{B}$ et $\gamma \in \mathcal{G}$ tels que $\beta = \gamma$.
- Calculer l'intersection de ces ensembles.
- Complexité :

Les algorithmes «génériques»

Les algorithmes les moins spécifiques s'appliquent à tous les groupes.

Énoncé général : $G = \langle g \rangle$ connu, $\#G = n$; $a = g^x$, trouver x.

Exemple des « pas de bébé, pas de géant » :

- Soit $C = \lceil \sqrt{n} \rceil$. x est inconnu, mais on sait qu'il existe une écriture $x = Cy + z$, avec $y, z \leq C$.
- Soit $\mathcal{B} = \{1, g, g^2, \ldots, g^C\}$, et $\mathcal{G} = \{ag^{-C}, ag^{-2C}, \ldots, ag^{-C^2}\}$.
- Il existe $\beta \in \mathcal{B}$ et $\gamma \in \mathcal{G}$ tels que $\beta = \gamma$.
- Calculer l'intersection de ces ensembles.
- Complexité : $O(C)$ en temps, et

Les algorithmes «génériques»

Les algorithmes les moins spécifiques s'appliquent à tous les groupes.

Énoncé général : $G = \langle g \rangle$ connu, $\#G = n$; $a = g^x$, trouver x.

Exemple des « pas de bébé, pas de géant » :

- Soit $C = \lceil \sqrt{n} \rceil$. x est inconnu, mais on sait qu'il existe une écriture $x = Cy + z$, avec $y, z \leq C$.
- Soit $\mathcal{B} = \{1, g, g^2, \ldots, g^C\}$, et $\mathcal{G} = \{ag^{-C}, ag^{-2C}, \ldots, ag^{-C^2}\}$.
- Il existe $\beta \in \mathcal{B}$ et $\gamma \in \mathcal{G}$ tels que $\beta = \gamma$.
- Calculer l'intersection de ces ensembles.
- Complexité : $O(C)$ en temps, et $O(C)$ en mémoire.

Ce qui compte c'est le terme principal : on retient : $O(\sqrt{n})$.

Il existe des algorithmes en $O(\sqrt{n})$ en temps, et $O(1)$ en mémoire.

Théorème : aucun algorithme générique ne peut faire mieux.

S'il n'y a pas mieux

Si pour une famille de groupes, seuls les algos génériques s'appliquent :

- Alors pour $\#G = n = 2^m$, il faut $O(2^{m/2})$ pour un log. di.
- Pour imposer 2^{128} comme boulot à l'attaquant, il faut au moins $n \approx 2^{256}$. Sous réserve qu'il n'y ait pas de meilleur algorithme !

Corps finis

Un objet mathématique dont vous connaissez un exemple : $\mathbb{Z}/p\mathbb{Z}$.

- $\mathbb{Z}/p\mathbb{Z}$ est un corps.
- Les éléments non nuls forment un groupe.
- Ce n'est qu'un cas particulier de corps fini (mais je ne vous ennuie pas avec les autres).

Avantages : • C'est plus compréhensible que la suite !
Inconvénients : • Dommage, ICI le pb du log. di. est difficile mais pas plus que la factorisation.

Coût du calcul de log discret dans $(\mathbb{Z}/p\mathbb{Z})^*$:

- $\exp(O((\log p)^{1/3}))$.
- Algorithme du crible algébrique (NFS), vous vous souvenez de lui ?
- Record de calcul connu aujourd'hui (2020) : 795 bits. (idem RSA !)

Courbes elliptiques

Courbes elliptiques (ECC) sur les corps finis :

- Comprendre ce qu'il se passe demande un bagage mathématique.
- Mais très simple au final, à la fois en matériel et logiciel.
- ICI Log discret très difficile. L'espion a la vie rude : Aucun algorithme connu autre que les génériques.

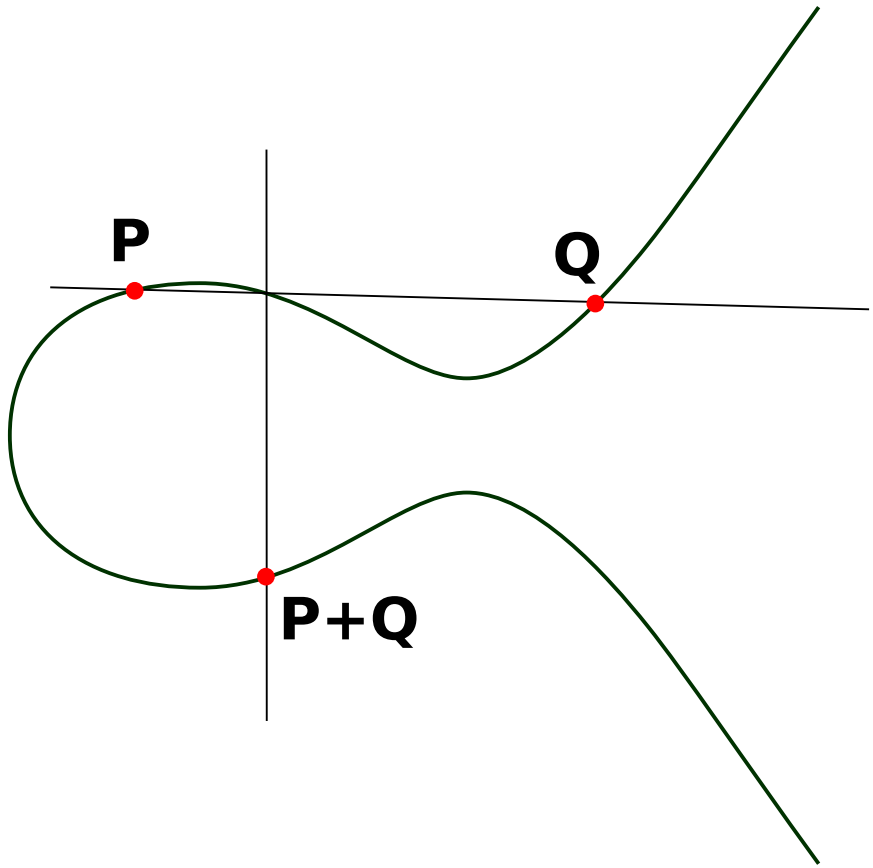
Une clé ECC : petit, mais costaud !

Pour imposer 2^{128} comme travail à l'attaquant :

- Clé ECC : 256 bits.
- Clé RSA : 3072 bits au moins.
- Clé DSA, avec $G = (\mathbb{Z}/p\mathbb{Z})^*$: 3072 bits au moins.

Loi de groupe

Courbe elliptique : solutions de $y^2 = x^3 + ax + b$ sur $\mathbb{Z}/p\mathbb{Z}$.



Attention, on note additivement !

Une clé RSA

```
AAAAB3NzaC1yc2EAAAADAQABAAACAQCx1mguXe0GfGySnV0/CXb5PYPUwgkF
/ZRoLsbwENh8SQFXHYgC4PjxC3zlW5VowQQRgacUvT3Vfed837VPahmjjThR
6VkeoqB34yU2I06H5s6UXa4+epiKcNQNL2+7ePkGZZSPtwTMohdxMPSy17kf
IsmvaXMrnWSyjo5ggimoMAbENrF4EtVgDP8STPKDvVNuB2dEGBjJKDME2HZ3
dVX6rGv2CU2njajXNPalhZd1mCC+CBv5+67tgGPbWXT8x0JXqWso4qrSL+py
hAkq65R2HIdnUHBWPZrq7oxIV1Gp4DbRR70sb9xdbdTcxZbh5Xf/Y0lnHYY2
LaAzXCE8rKuUMiKISCYydBPGPjUsPuFbAsz6qOZ/lQTO5opK7vlmjLxfAhuo
uoymV5laqM0SonBCY8T2QUkG8iAt8ckn35QSS8OI48VuTWcgT6K2DmzXeuQ4
VnULp+h0VlLByqjrJbK8eRfbH2gloqwfJbUWJUkdjPElLlQygnMAOl8YcDTL
G9W1iuMbcSAVtZ3Qx+9bGUmibR2YVJyhhiWS+zM/VwhkEfmRQng7BepYg3jn
fQn90OTjRm2uj93CQkMvvQhMglZEIFCPCGOmJc7PBwItp7HczUrKzEBYJ3IR
mLYZFZkHyXmlA/bvy3LCOszah1sDie0uiIGIuJocaAH6skkkTrxbqw==
```

Une clé ECC

```
AAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBB7SKdUBQ
oNt8sHcrafwO2S/X/ojzuEHSEXJOAjiQRz7Tieczf+TV9oCadjkyDBTYZ5zq
TCbelHILDJNCp+aVOI=
```

Résumé

Beaucoup de choix à faire.

- Objectifs : confidentialité, authenticité, intégrité.
- Primitives : Échange de clé, Chiffrement, Signature.
- Algorithmes : clé privée, clé publique, hachage.

Performances

Performances de chiffrement : pour un niveau de sécurité 2^{128}.

- Clé privée, AES-128 : ≈ 1 Go/s sur un cœur.
- Clé publique, RSA-3072 : 20 Mo/s (chiffrement $e = 3$), 50 Ko/s (signature).
- Clé publique, ECC-256 : 400 Ko/s (chiffrement ou signature).

Moralité :

- Échange de clés pour fabriquer une clé de session, puis AES.
- La pression en terme d'efficacité reste souvent sur le symétrique : quelques octets chiffrés en asymétrique, puis quelques mégaoctets avec AES.