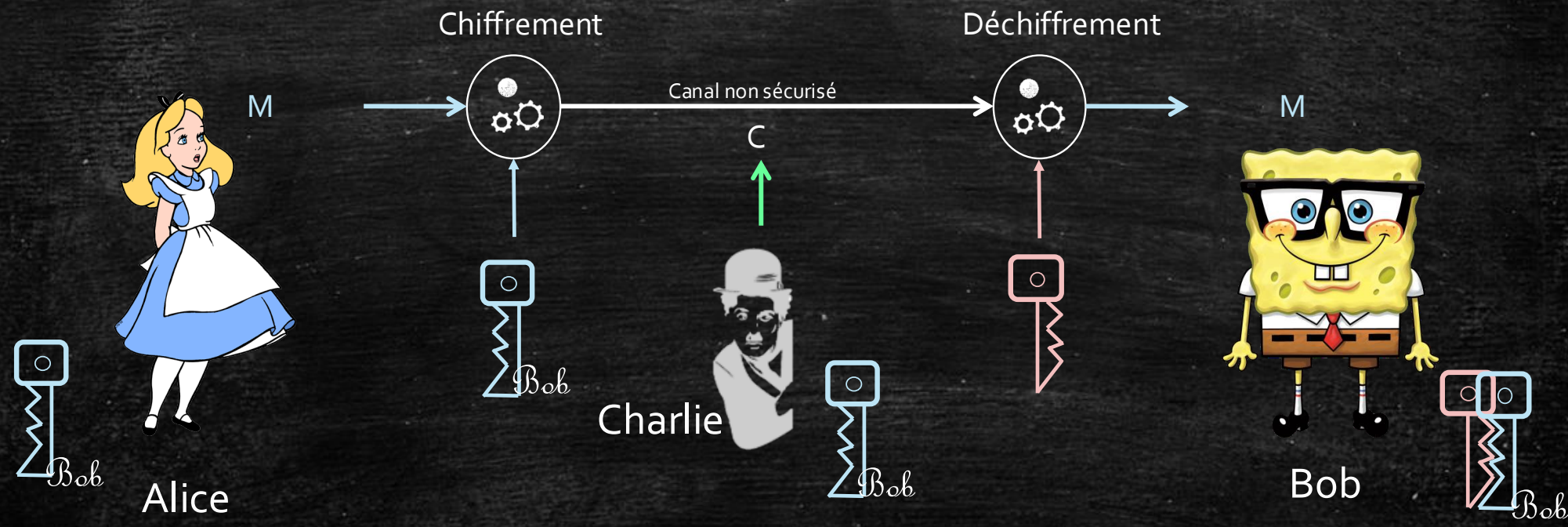
Introduction à la cryptographie

Louiza Khati

4A-Partie 2

Chiffrement asymétrique



Alice envoie un message à Bob (utilisation du bi-clé de Bob)

Chiffrement asymétrique

- Propriétés sur les clés
  - La connaissance de la clé publique **ne doit pas permettre** de retrouver la clé privée
  - La clé privée et la clé publique sont **liées**
- Propriétés sur le schéma : fonction à sens unique
  - **Chiffrer** un message doit être facile
  - **Déchiffrer** un message **sans la clé** doit être **très difficile!**
- Repose sur un problème difficile (preuve par réduction)
  - La **factorisation** d'entiers
  - Le **logarithme** discret

Chiffrement asymétrique : Factorisation

- **Factorisation** des nombre entiers
  - 53 x 37 = ?
  - 1403 = ?

Chiffrement asymétrique : Factorisation

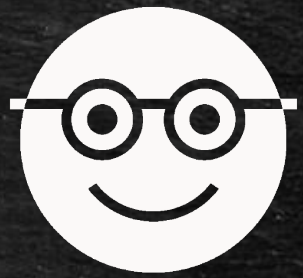
- **Factorisation** des nombre entiers
  - 53 x 37 = 1961
  - 1403 = 61 * 23

Chiffrement asymétrique : Factorisation

- **Factorisation** des nombre entiers
  - 53 x 37 = 1961
  - 1403 = 61 * 23
- Et
  25195908475657893494027183240048398571429282126204032027777137836043662020707595555626401852588078440691829064124951508218929855914917618450280848912007284499268739280728777673597141834727026189637501497182469116507761337985909570009733045974880842840179742910064245869181719511874612151517265463228221686998754918242243363725908514186546204357679842338718477444792073993423658482382428119816381501067481045166037730605620161967625613384414360383390441495263443219011465754445417842402092461651572335077870774981712577246796292638635637328991215483143816789988504044536402352738195137863656439121201039712282212072035 ????

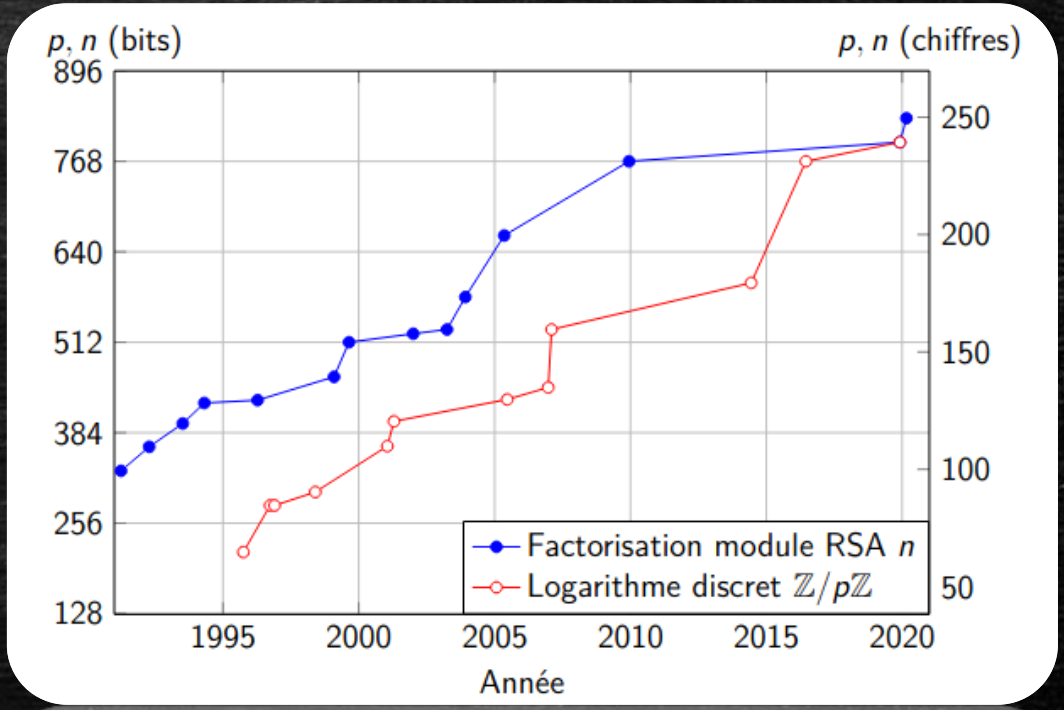
Complexité : produit et factorisation

- Produit de deux nombres de n bits
- Coût d'un produit : $N_1 \times N_2$
  - Soient N_1 et N_2 deux nombres premiers sur n bits
  - Méthode naïve : coût $O(n^2)$
  - Méthodes plus efficaces : complexité **quasi-linéaire**
- Factorisation
  - Pour un entier sur N sur n bits, complexité **exponentielle**
  - Algorithme naïf : $O(\sqrt{N}) \rightarrow O(2^{n/2})$



Factorisation

- Record de factorisation
  - 768 bits et 232 chiffres décimaux



https://members.loria.fr/AGuillevic/files/teaching/NFS/techniques-de-l-ingenieur-record-calcul-RSA240.pdf

Rappels : Algorithme d'Euclide

- Soient a et b deux entiers : pgcd(a,b) = pgcd(b, r) où r = a mod(b)
- Exemple : pgcd (119, 91) = ?
  - 119 = 1* 91 + 28
  - 91 = 3*28 + 7
  - 28 = 4* 7 + 0

PGCD!

Rappels : Algorithme d'Euclide

- Soient a et b deux entiers : pgcd(a,b) = pgcd(b, r) où r = a mod(b)
- Exemple : pgcd (119, 91) = 7
  - 119 = 1* 91 + 28
  - 91 = 3*28 + 7
  - 28 = 4* 7 + 0

Rappels : Théorème de Bezout

- Soient a et b deux entiers naturels tel que pgcd(a,b) = d alors il existe deux entiers relatifs u et v tel que :

$$d = a*u+b*v$$

u et v sont appelés les coefficients de Bezout.

- Exemple : a = 21 et b = 12 → d = 3
  - (-1)*21 + 2 * 12 = 3 → u = -1 et v = 2
- Cas particulier d = 1 : a et b premiers entre eux alors il existe u et v tel que

$$1 = a*u+b*v$$

(calcul inverse modulaire)

Rappels : Algorithme d'Euclide étendu

- Version récursive de l'algorithme d'Euclide
- Permet de trouver les coefficients de Bezout
- Exemple : pgcd (119, 91) = ?
  - (1) 119 = 1* 91 + 28
  - (2) 91 = 3*28 + 7
  - (3) 28 = 4* 7 + 0
- Reconstruction (trouver les coefficients de Bezout) :
  - (2)→ 91 – 3*28 = 7
  - Avec (1) →91 – 3 * (119 – 1*91) = 7
  - Finalement 4*91 – 3* 119 = 7

Rappels : Indicatrice d'Euler

- Fonction qui à tout entier naturel N non nul associe le nombre d'entiers compris entre 1 et N (inclus) et premiers avec N.

- Exemples :
  - $\phi(5) = 4$ ($\{1,2,3,4\}$)
  - Si p est premier : $\phi(p) = p-1$
  - Si p_i premier et $n = \Pi\, p_i$ alors $\phi(n) = \Pi\,(p_i-1)$
- Remarque : Pour calculer $\phi(n)$, il faut connaitre la décomposition en facteurs premiers de n!

Indicatrice d'Euler : Exemples

- n = 17 alors $\phi(n) = ?$ Ensemble premier avec n ?
- n = 15 alors $\phi(n) = ?$ Ensemble premier avec n ?
- n = 113 alors $\phi(n) = ?$ Ensemble premier avec n ?
- n = 42 alors $\phi(n) = ?$ Ensemble premier avec n ?

- Si n = pq avec p et q deux grands nombres premiers alors ?

Rappels(?) : Théorème d'Euler

- Soient a et N deux entiers
- Théorème d'Euler : si pgcd(a,N)=1 alors

$$a^{\phi(N)} = 1 \bmod (N)$$

- Cas particuliers : soient p et q deux entiers premiers distincts, N=pq et a entier relatif tel que pgcd(a,N) = 1 alors

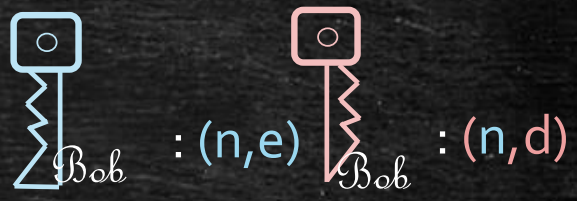
$$a^{(p-1)(q-1)} = 1 \bmod (N)$$

RSA

- Présenté en 1977
- Rivest, **S**hamir et **A**dleman (RSA)
- Basé sur la problème de la **factorisation**
- Mécanismes basés sur le cryptosystème RSA :
  - Chiffrement RSA (confidentialité des données)
  - Signature RSA (intégrité des données + non répudiation)

RSA : génération des clés

- Génération du bi-clé de chiffrement de Bob
  - Tirer aléatoirement deux grands nombres entiers p et q
    - Certaines propriétés doivent être vérifiées.
  - Calculer le module N = p q
    - Le module N est un paramètre public
  - Calculer l'indicatrice d'Euler : $\phi(N) = (p-1)(q-1)$
  - Choisir l'exposant public e tel que PGCD(e, $\phi(N)$) = 1
  - Calculer l'exposant privé d tel que $d\,e = 1 \bmod(\phi(N))$
    - Algorithme d'Euclide étendu



RSA : génération des clés

- Génération du bi-clé de chiffrement de Bob
  - Tirer aléatoirement deux grands nombres entiers p et q
    - Certaines propriétés doivent être vérifiées.
  - Calculer le module N = p q
    - Le module N est un paramètre public
  - Calculer l'indicatrice d'Euler : $\phi(N) = (p-1)(q-1)$
  - Choisir l'exposant public e tel que PGCD(e, $\phi(N)$) = 1
  - Calculer l'exposant privé d tel que d e= 1 mod($\phi(N)$)

```
khati@khati-ThinkPad-X280: $ time  openssl genrsa -out mykey.pem 1024

real    0m0,033s
user    0m0,028s
sys     0m0,005s
khati@khati-ThinkPad-X280: $ time  openssl genrsa -out mykey.pem 2048

real    0m0,250s
user    0m0,242s
sys     0m0,009s
khati@khati-ThinkPad-X280: $ time  openssl genrsa -out mykey.pem 3072

real    0m0,968s
user    0m0,959s
sys     0m0,009s
khati@khati-ThinkPad-X280: $ time  openssl genrsa -out mykey.pem 4096

real    0m2,330s
user    0m2,318s
sys     0m0,013s
```

- Taille de la clé :
  - [ANSSI] Module, exposant secret d : **2048 bits** minimum (→2030) ensuite 4096
  - [ANSSI] Exposant public e : > 2^{16} = 65536 (exemple : e = 65537)

RSA : génération des clés

- Génération du bi-clé de chiffrement de Bob
  - Tirer aléatoirement deux grands nombres entiers p et q
    - Certaines propriétés doivent être vérifiées.
  - Calculer le module N = p q
    - Le module N est un paramètre public
  - Calculer l'indicatrice d'Euler : $\phi(N) = (p-1)(q-1)$
  - Choisir l'exposant public e tel que PGCD(e, $\phi(N)$) = 1
  - Calculer l'exposant privé d tel que $d\,e = 1 \bmod(\phi(N))$
    - Calculé grâce à l'algorithme d'Euclide étendu
    - Calcul de d : il existe u et v tel que $u*e+v*\phi(n) = 1 \rightarrow u*e = 1 \bmod(\phi(N))$ [Théorème de Bezout]

Exemple simple

p = 17 et q = 31

N = ?

$\phi(N)$ = ?

e = 7 , pgcd(7,480)=1

d = ?

RSA : génération des clés

- Génération du bi-clé de chiffrement de Bob
  - Tirer aléatoirement deux grands nombres entiers p et q
    - Certaines propriétés doivent être vérifiées.
  - Calculer le module N = p q (pas trop proches!)
    - Le module N est un paramètre public
  - Calculer l'indicatrice d'Euler : $\phi(N) = (p-1)(q-1)$
  - Choisir l'exposant public e tel que PGCD(e, $\phi(N)$) = 1
  - Calculer l'exposant privé d tel que $d\,e = 1 \bmod(\phi(N))$
    - Calculé grâce à l'algorithme d'Euclide étendu
    - Calcul de d : il existe u et v tel que $u*e+v*\phi(n) = 1 \rightarrow u*e = 1 \bmod(\phi(N))$ [Théorème de Bezout]

Exemple simple

p = 17 et q = 31

N = 17 *31 = 527

$\phi(N) = (17-1)(31-1) = 480$

e = 7 , pgcd(7,480)=1

d = ?

RSA : génération des clés

- Génération des clés
  - Calculer d tel que pgcd(7,480)=1
  - Algorithme d'Euclide étendu

Exemple simple

480 = 68*7+4

7 = 1*4 + 3

4 = 1*3 + 1

1 =1 *1 + 0

1 = 2*480 + (-137) * 7

1 = -137 * 7 mod (480) et -137 = 343 mod (480)

→ 1 = 343 * 7 mod (480)

Clé privée d = 343

: (527,7)

: (527,343)

Exemple : Euclide étendu

- (104,53) → 53^{-1} = ? mod 104
- (31, 67) → d tel que d*31 mod 67
- (52,91) → d tel que d*52 mod 91

Rappels : Chiffrement asymétrique

Trois algorithmes : k = paramètre de sécurité = taille de la clé

- Génération de clé : (sk,pk) ← Keygen(k) avec sk = (d,N) et pk=(e,N)

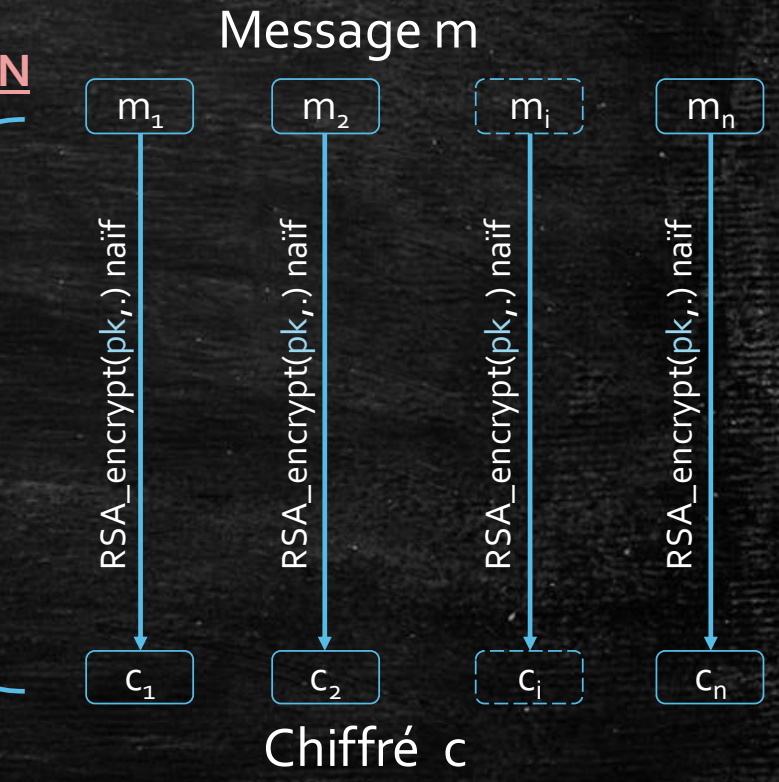
  RSA

- Chiffrement : c ← encrypt(pk, m)

- Déchiffrement : m ← decrypt(sk, c)

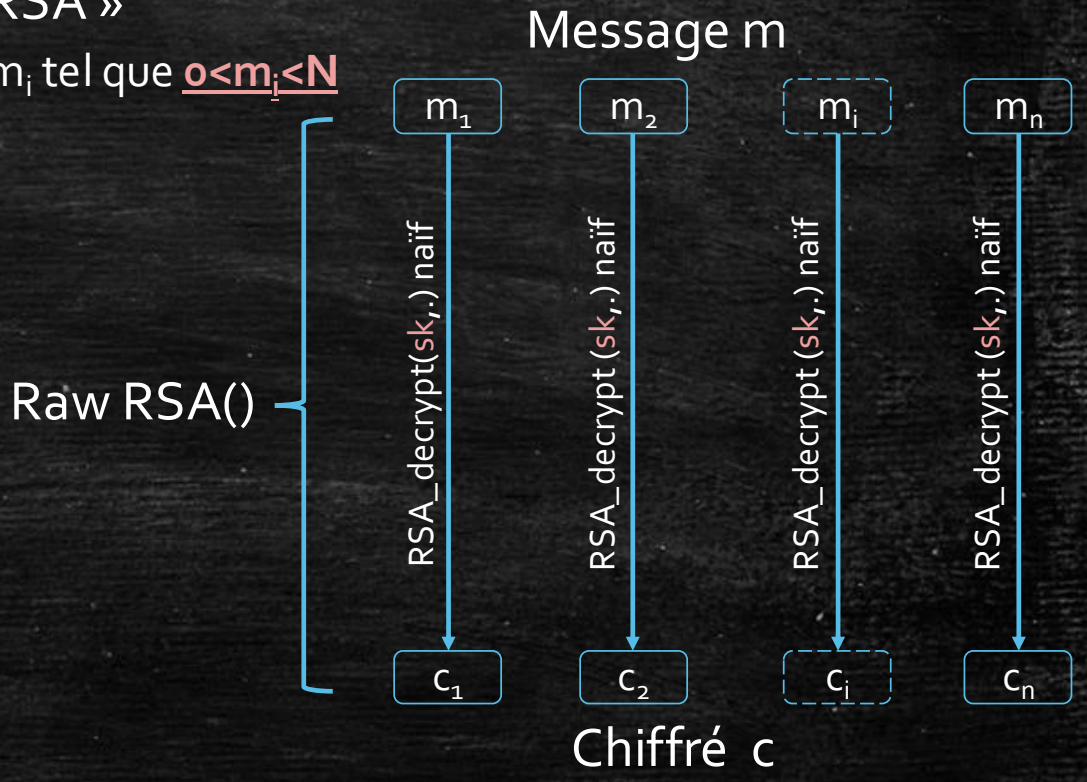
RSA : Chiffrement naïf

- Opération de chiffrement « Raw RSA »
  - Alice découpe le message m en bloc m_i tel que $0<m_i<N$
  - Chiffrement de chaque message m_i :
    - $c_i = m_i^{\,e} \bmod (N)$
  - Déchiffrement de c :
    - $c_i^{\,d} \bmod(N) = m_i$



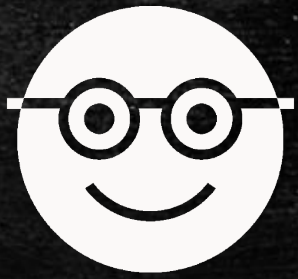
RSA : Chiffrement naïf

- Opération de chiffrement « Raw RSA »
  - Alice découpe le message m en bloc m_i tel que $0<m_i<N$
  - Chiffrement de chaque message m_i :
    - $c_i = m_i^{\,e} \bmod (N)$
  - Déchiffrement de c :
    - $c_i^{\,d} \bmod(N) = m_i$
-



Modèle de sécurité : chiffrement asymétrique

- Malléabilité
- Adversaire $\mathcal{A}$:
  - A la vue d'un chiffré C d'un message inconnu M (ou plusieurs), il peut construire un chiffré C* tel que la relation entre M* et M soit connue.



Chiffrement asymétrique : Sécurité

- RSA naïf :
  - Taille du message limité
  - Même avec l'hypothèse que le message m<N → chiffrement déterministe

- Raw RSA
  - Chiffrement déterministe
  - Adversaire : Possibilité de calculer le chiffré de n'importe quel message

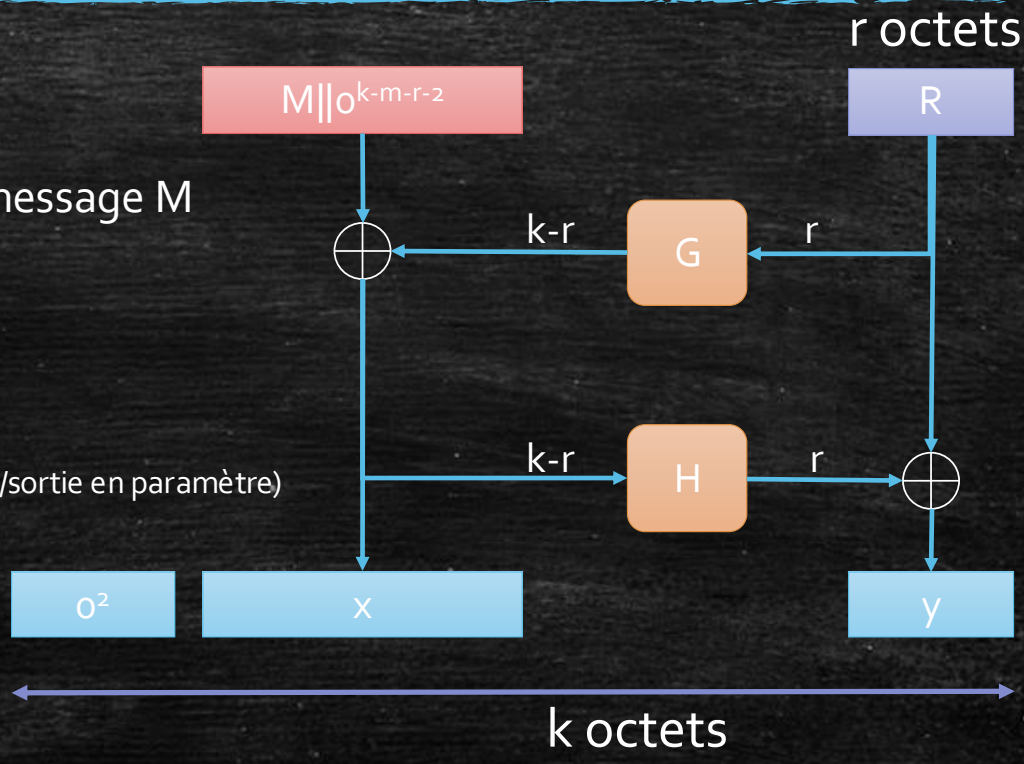
- Question : comment définir un chiffrement asymétrique robuste ?

Chiffrement : RSA-OAEP

- « Optimal Asymmetric Encryption Padding »
- Par Mihir Bellare et Phillip Rogaway en 1994
- Standard PKCS v2.2 (Plusieurs RFC : 2437, 3447, …)
- Ajouter de l'aléa :
  - Masquer le message M (1<M<N avec N le module RSA)
  - Rendre le chiffrement probabiliste
  - Taille de **l'aléa suffisamment grand** ?
    - Au moins 128 bits sinon recherche exhaustive sur cet aléa

Chiffrement : RSA-OAEP

- Encapsulation **avant chiffrement** du message M
  - De taille m octets
- O^i : i octet à zéro
- Module de n bits (k = n/8 octets)
- R valeur aléatoire sur r octets
- Fonctions G et H (oracles aléatoires) :
  - « MGF » Mask Generation Function (tailles entrée/sortie en paramètre)
  - Basé sur une fonction de hachage



Chiffrement : RSA-OAEP

- Preuve de sécurité : RSA-OAEP
  - Paramètres RSA problème difficile (**paramètres bien choisis**)

- Ne permet de chiffrer qu'un « bloc de message »

Chiffrement : RSA-OAEP

- Preuve de sécurité : RSA-OAEP robuste (IND-CCA).
  - Paramètres RSA problème difficile (**paramètres bien choisis**)

- Ne permet de **chiffrer qu'un « bloc de message »**
  - Chiffrement asymétrique très long !
  - A combiner avec un chiffrement symétrique
    - Chiffrer une clé symétrique K (petite taille) avec RSA-OAEP,
    - Chiffrer le message long avec la clé et le chiffrement symétrique.

Chiffrement : RSA-OAEP

- Preuve de sécurité : RSA-OAEP est robuste (IND-CCA2).
  - Paramètres RSA problème difficile (**paramètres bien choisis**)

- TP :
  - Attaques si les paramètres sont mal choisis

Crypto-système RSA

- Fonction à sens unique :
  - Calculer M^e mod(N) est facile
  - Sans la trappe d, à partir de $C = M^e$ mod(N), il est très difficile de trouver M
    = problème RSA
  - Repose sur le problème de la **factorisation**
- Avec des paramètres biens choisis :
  - Génération de p et q premiers
    - p et q de taille similaire
      - Recherche exhaustive
  - Exposant public e pas trop petit ($>2^{16}$)
  - Exposant secret d de même taille que le module
  - Taille de module **au moins 2048 bits**.

Conclusion : Chiffrement asymétrique

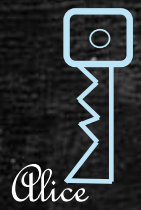
- Permet de chiffrer que de « petites données »
- En pratique : permet de chiffrer une clé symétrique K dans le but de chiffrer les données avec K.
- Remarque : nous avons vu des chiffrements asymétriques basés sur RSA mais il en existe d'autres!

Conclusion : Chiffrement asymétrique

Bob veut envoyer des données chiffrées à Alice.

K « petite »
(128 à 512 bits)

Données volumineuses
Plusieurs GO



Chiffrement asymétrique

lent

K_chiffree

Chiffrement symétrique

rapide

Grosses_donnees_chiffree

Allier les avantages du chiffrement symétrique et asymétrique!

Signature

Exemples basés sur RSA

Signature manuscrite

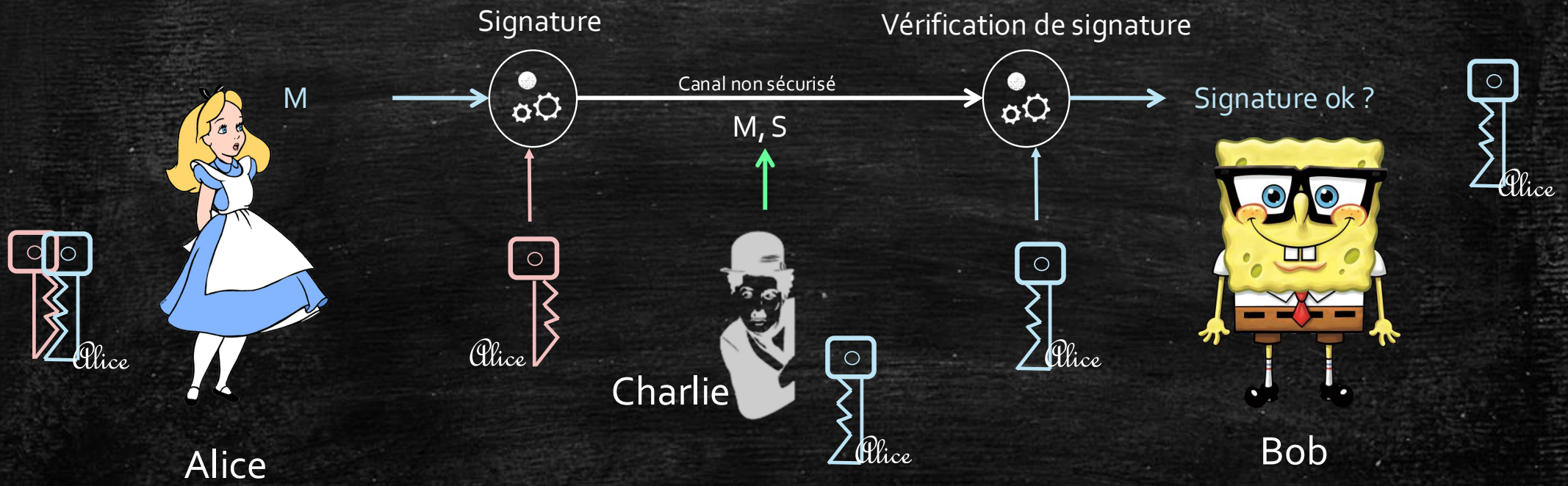
- engage la **responsabilité du signataire**
- **physiquement attachée** au document signé
- **vérification** de la signature **par comparaison** avec une signature précédente

Signatures électroniques

- Elles garantissent :
  - **l'intégrité** du document
  - l'authentification de l'émetteur
  - la **non-répudiation**

- Elles dépendent :
  - du message
  - du signataire identifié par sa paire de clés publique/privée

Signature asymétrique



Alice envoie un message signé à Bob (utilisation de la bi-clé d'Alice)
Différent du chiffrement asymétrique!

Signature

- Algorithme asymétrique sinon pas de **non-répudiation**
  - Si algo symétrique → clé partagée K
- Seul le détenteur de la **clé privée** peut **signer**

- Tout **le monde** peut **vérifier** la signature (clé publique)

- Une signature peut être **rejouée**

Signature : attaquant

- But ?
- Fournir une **forge/contrefaçon** (m^*, s^*) pour un utilisateur tel que :
  - (m^*, s^*) n'a jamais été généré par l'utilisateur
- Exemple : Alice génère plusieurs signatures (m_1,s_1), (m_2,s_2),…, (m_q,s_q)
  - Un attaquant peut fournir (m_3,s_3) → ce n'est pas une forge
  - Si un attaquant fournit $(m^*, s^*) \neq (m_i,s_i)$ pour tout $i \in \{1,\ldots,q\}$, c'est une forge.

  → un couple valide (m^*, s^*) non générée par Alice.