Cryptographie

Cryptographie à clé publique et RSA

Léo Colisson Palais
Master CSI 2024 – 2025

leo.colisson-palais@univ-grenoble-alpes.fr

https://leo.colisson.me/teaching.html

Cryptographie à clé publique : Motivations

Rappel: Cryptographie symétrique vs asymétrique

Chiffrement asymétrique

😀 Pas besoin d'échanger de secrets (e.g. internet)

😰 Hypothèses plus fortes
factorisation, LWE…
(fonctions très structurées)

😰 Moins efficace

😰 Pas de sécurité statistique

Chiffrement symétrique

😰 Doit partager des secrets

😀 Plus faibles hypothèses
(moins de structure)

😀 Plus efficace

😀 Sécurité statistique possible
(mais pas pratique)

$\Rightarrow$ Systèmes hybrides: **combiner les deux** = meilleur des deux mondes
(efficace + pas de secret à distribuer)

Rappel: Cryptographie symétrique vs asymétrique

Ce cours

Chiffrement asymétrique

- 😀 Pas besoin d'échanger de secrets (e.g. internet)
- 😰 Hypothèses plus fortes factorisation, LWE… (fonctions très structurées)
- 😰 Moins efficace
- 😰 Pas de sécurité statistique

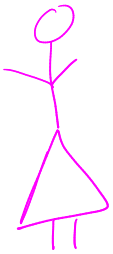
Chiffrement symétrique

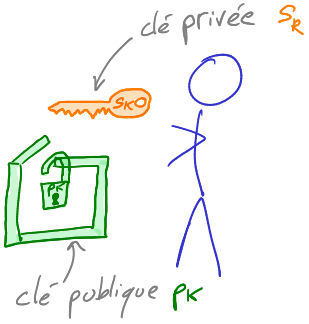
- 😰 Doit partager des secrets
- 😀 Plus faibles hypothèses (moins de structure)
- 😀 Plus efficace
- 😀 Sécurité statistique possible (mais pas pratique)

⇒ Systèmes hybrides: **combiner les deux** = meilleur des deux mondes (efficace + pas de secret à distribuer)

Cryptographie asymétrique

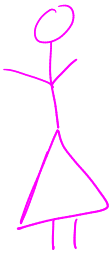
1. Bob génère une clé publique pk et secrète sk :

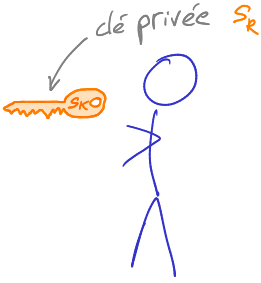




Cryptographie asymétrique

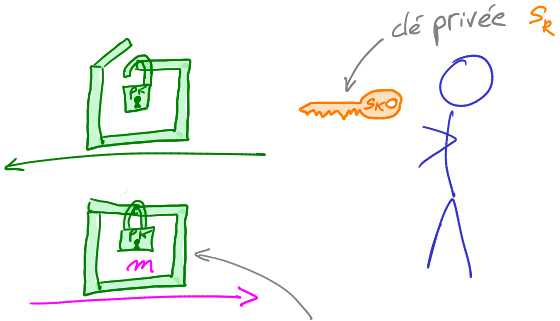
1. Bob génère une clé publique p_K et secrète s_K :
2. Bob distribue p_K publiquement (en pratique via PKI)





Cryptographie asymétrique

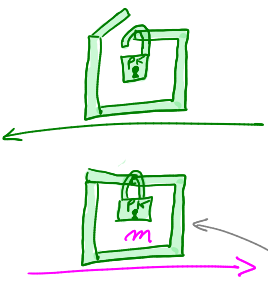
1. Bob génère une clé publique pk et secrète sk :
2. Bob distribue pk publiquement (en pratique via PKI)

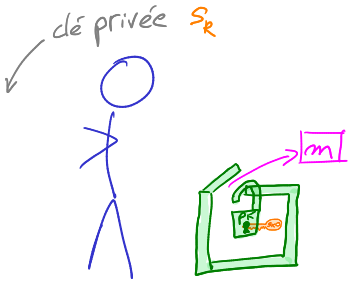


3. Alice chiffre m : $Enc_{pk}(m)$ et l'envoie à Bob

Cryptographie asymétrique

1. Bob génère une clé publique pk et secrète sk :
2. Bob distribue pk publiquement (en pratique via PKI)





3. Alice chiffre m : $Enc_{pk}(m)$ et l'envoie à Bob
4. Bob déchiffre avec sk et retrouve m

Comment obtenir cryptographie asymétrique

Crypto asymétrique fondamentalement + puissante que crypto symétrique :
$\Rightarrow$ **impossible de se baser (seulement) sur PRF/hash/block-cipher…**

Hypothèses en plus nécessaires :

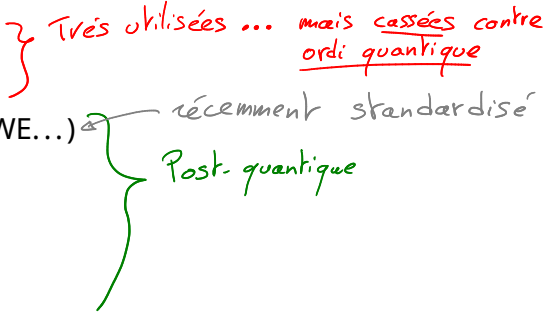
- Difficulté de factoriser/RSA
- Courbes elliptiques
- Réseaux euclidiens (Lattices, LWE…)
- Codes
- Isogénies
- Polynômes multivariés
- …

Comment obtenir cryptographie asymétrique

Crypto asymétrique fondamentalement + puissante que crypto symétrique :
$\Rightarrow$ **impossible de se baser (seulement) sur PRF/hash/block-cipher…**

Hypothèses en plus nécessaires :

- Difficulté de factoriser/RSA
- Courbes elliptiques
- Réseaux euclidiens (Lattices, LWE…)
- Codes
- Isogénies
- Polynômes multivariés
- …

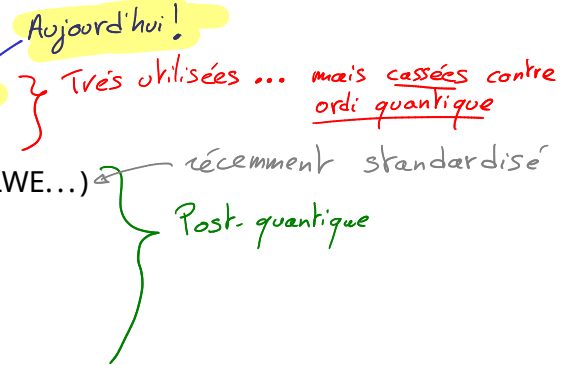


Comment obtenir cryptographie asymétrique

Crypto asymétrique fondamentalement + puissante que crypto symétrique :
$\Rightarrow$ **impossible de se baser (seulement) sur PRF/hash/block-cipher…**

Hypothèses en plus nécessaires :

- Difficulté de factoriser/RSA
- Courbes elliptiques
- Réseaux euclidiens (Lattices, LWE…)
- Codes
- Isogénies
- Polynômes multivariés
- …



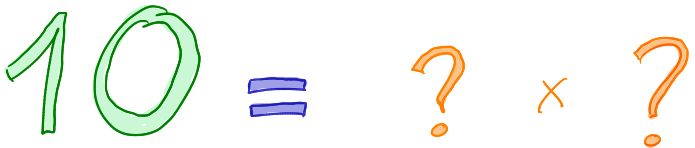
Rivest, Shamir, Adleman, 1977 : article célébrissime

↓

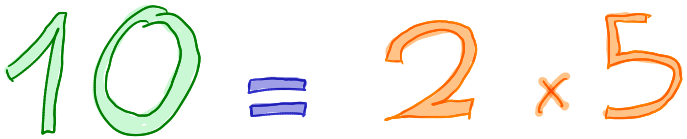
RSA

Le problème de la factorisation :

Le problème de la factorisation :



Le problème de la factorisation :



Le problème de la factorisation :

Le problème de la factorisation :

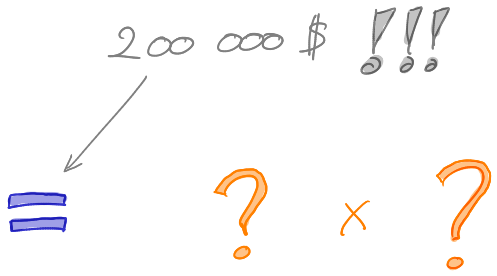
RSA

Le problème de la factorisation :

25195908475657893494027183240048398571429282126204032027777137836043662020707595556264018525880784406918290641249515082189298559149176184502808489120072844992687392807287776735971418347270261896375014971824691165077613379859095700097330459748808428401797429100642458691817195118746121515172654632282216869987549182422433637259085141865462043576798423387184774479207399342365848238242811981638150106748104516603773060562016196762561338441436038339044149526344321901146575444541784240209246165157233507787077498171257724679629263863563732899121548314381678998850404453640235273819513786365643912120103971228221207203571


RSA

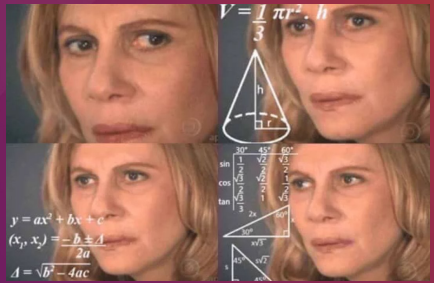
Le problème de la factorisation :

25195908475657893494027183240048398571429282126204032027777137836043662020707595556264018525880784406918290641249515082189298559149176184502808489120072844992687392807287776735971418347270261896375014971824691165077613379859095700097330459748808428401797429100642458691817195118746121515172654632282216869987549182422433637259085141865462043576798423387184774479207399342365848238242811981638150106748104516603773060562016196762561338441436038339044149526344321901146575444541784240209246165157233507787077498171257724679629263863563732899121548314381678998850404453640235273819513786365643912120103971228221207203571


?

Comment utiliser ce problème pour **faire de la crypto** ?

Pré-requis calculatoires



Pré-requis calculatoires

$a^0 = ?$

A. 0
B. 1
C. a

Pré-requis calculatoires

$a^0 = ?$

A. 0

B. 1 ✓

C. a

Pré-requis calculatoires

$a^{b+c} = ?$

A. $a^b + a^c$
B. $a^b a^c$
C. $(a^b)^c$
D. a^{b^c}
E. Aucun

Pré-requis calculatoires

$a^{b+c} = ?$

A. $a^b + a^c$

B. $a^b a^c$ ✔ Si doute : essayez avec nombres:
   - $1^{0+0} = 1^0 = 1 = 1^0 \times 1^0$ (invalide la première formule, ça donne $1^0 + 1^0 = 2$, mais pas la dernière)
   - ou encore $3^{1+2} = 3^3 = 27$, $3^1 3^2 = 3 \times 9 = 27$

C. $(a^b)^c$

D. a^{b^c}

E. Aucun

Pré-requis calculatoires

$a^{bc} = ?$

A. $a^b + a^c$
B. $a^b a^c$
C. $(a^b)^c$
D. a^{b^c}
E. Aucun

Pré-requis calculatoires

$a^{bc} = ?$

A. $a^b + a^c$

B. $a^b a^c$

C. $(a^b)^c$ ✓

D. a^{b^c}

E. Aucun

Pré-requis calculatoires

$a^{-b} = ?$

A. $-a^b$

B. $a - b$

C. $-b \log a$

D. $\frac{1}{a^b}$

E. Aucun

Pré-requis calculatoires

$a^{-b} = ?$

- A $-a^b$
- B $a - b$
- C $-b \log a$
- D $\frac{1}{a^b}$ ✓
- E Aucun

Pré-requis calculatoires

$aa^{-1} = ?$

A. 0
B. 1
C. a
D. Autre

Pré-requis calculatoires

$aa^{-1} = ?$

A 0

B 1 ✓ C'est même la définition d'un inverse

C a

D Autre

Pré-requis calculatoires

$16 \bmod 10$

A. 0
B. 1
C. 6
D. 10
E. Autre

Pré-requis calculatoires

$16 \bmod 10$

A. 0
B. 1
C. 6 ✓
D. 10
E. Autre

Pré-requis calculatoires

?

$$((a \bmod c) + (b \bmod c)) \bmod c \stackrel{?}{=} (a + b) \bmod c$$

A. Vrai

B. Faux

Pré-requis calculatoires

$$((a \bmod c) + (b \bmod c)) \bmod c \stackrel{?}{=} (a + b) \bmod c$$

A Vrai ✓

B Faux

Pré-requis calculatoires

?

$((a \bmod c)(b \bmod c)) \bmod c \stackrel{?}{=} (ab) \bmod c$

A Vrai

B Faux

Pré-requis calculatoires

$((a \bmod c)(b \bmod c)) \bmod c \stackrel{?}{=} (ab) \bmod c$

A. Vrai ✓

B. Faux

Pré-requis calculatoires

$3^{-1} \bmod 11 = ?$

- A 0
- B 0.33…
- C 1
- D 3
- E 4
- F 11
- G Autre

Pré-requis calculatoires

$3^{-1} \bmod 11 = ?$

A. 0

B. 0.33…

C. 1

D. 3

E. 4 ✔ L'inverse de 3 est x tel que $3x = 1 \bmod 10$. Méthode brute :

- $3 \times 1 = 3 \neq 1 \bmod 11$
- $3 \times 2 = 6 \neq 1 \bmod 11$
- $3 \times 3 = 9 \neq 1 \bmod 11$
- $3 \times 4 = 12 = 1 \bmod 11 \Rightarrow 3^{-1} = 4$!

F. 11

G. Autre

Pré-requis calculatoires

$3^{-2} \bmod 11 = ?$

A. 0
B. 1
C. 4
D. 8
E. 11
F. Autre

Pré-requis calculatoires

$3^{-2} \bmod 11 = ?$

- A 0
- B 1
- C 4
- D 8
- E 11
- F Autre ✔ C'est 5:

$$3^{-2} = 3^{-1 \times 2} = (3^{-1})^2 = 4^2 = 16 \bmod 11 = 5 \bmod 11$$

Pré-requis calculatoires

?

$3^{-2} \bmod 11 = ?$

- A 0
- B 1
- C 4
- D 8
- E 11
- F Autre ✔ C'est 5:

Formule vue avant (puissances)

$$3^{-2} = 3^{-1 \times 2} = (3^{-1})^2 = 4^2 = 16 \bmod 11 = 5 \bmod 11$$

Pré-requis calculatoires

$3^{-2} \bmod 11 = ?$

?

- A 0
- B 1
- C 4
- D 8
- E 11
- F Autre ✔ C'est 5:

Formule vue avant (puissances)

Question précédente

$$3^{-2} = 3^{-1 \times 2} = (3^{-1})^2 = 4^2 = 16 \bmod 11 = 5 \bmod 11$$

Pré-requis calculatoires

?

Est il toujours possible de calculer un inverse modulaire (e.g. $x^{-1} \bmod n$) ?

A Oui

B Non

Pré-requis calculatoires

?

Est il toujours possible de calculer un inverse modulaire (e.g. $x^{-1} \bmod n$) ?

A. Oui

B. Non ✓ Non, ce n'est possible que si pgcd$(x, n) = 1$ (cf. algo plus tard)

Pré-requis calculatoires

?

Le calcul d'inverse modulaire (e.g. $x^{-1} \bmod n$) est il efficace à calculer ? Si oui comment ?

A Oui

B Non

Pré-requis calculatoires

Le calcul d'inverse modulaire (e.g. $x^{-1} \bmod n$) est il efficace à calculer ? Si oui comment ?

A Oui ✓ Algorithme **d'Euclide Étendu** pour trouver u, v tq $xu + nv = 1$ (possible ssi pgcd$(x, n) = 1$). Exemple pour $3^{-1} \bmod 11$:

inverse car $xu = 1 - nv$ $= 1 \bmod n$

Relation Bezout

(a) $11 = 3 \times (3) + 2$ → Division euclidienne de 11 par 3

(b) $3 = 2 \times (1) + 1$

on boucle jusqu'à avoir 1 (pgcd(x,n)=1)

?

On remonte :

$1 = 3 - 2 \times (1)$

via (b)

$= 3 - (11 - 3 \times (3)) \times (1)$

via (a), $2 = 11 - 3(3)$

Simplifie → $= 11 \times (-1) + 3 \times (4)$

on remonte équation par équation : si équation est $a = b(c) + d$, on utilise $d = a - b(c)$ pour remplacer les d par des a & b.

Donc $3 \times 4 = 1 - 11 \times (-1)$, i.e.

inverse ! car

$3 \times 4 \bmod 11 = 1 - 11 \times (-1) \bmod 11 = 1$

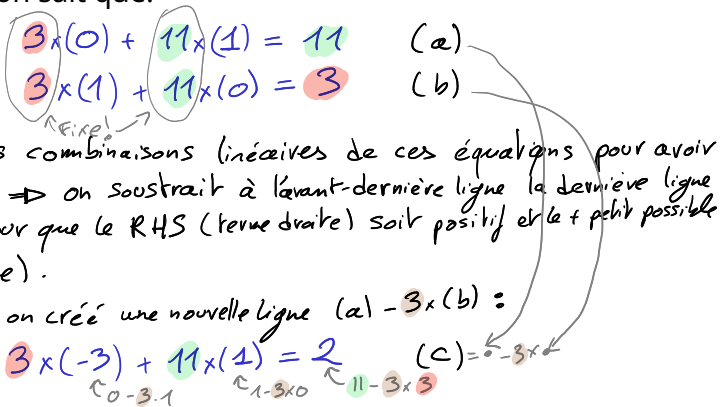
Pré-requis calculatoires

Autre méthode (plusieurs autres solutions) : on souhaite trouver u, v tels que $3u + 11v = 1$. Au début, on sait que:

$$3 \times (0) + 11 \times (1) = 11$$
$$3 \times (1) + 11 \times (0) = 3$$

Pré-requis calculatoires

Autre méthode (plusieurs autres solutions) : on souhaite trouver u, v tels que $3u + 11v = 1$. Au début, on sait que:



Pré-requis calculatoires

Autre méthode (plusieurs autres solutions) : on souhaite trouver u, v tels que $3u + 11v = 1$. Au début, on sait que:

$3 \times (0) + 11 \times (1) = 11$ (a)

$3 \times (1) + 11 \times (0) = 3$ (b)

Fixe!

On va alors faire des combinaisons linéaires de ces équations pour avoir $3 \times (?) + 11 \times (?) = 1$ ⇒ on soustrait à l'avant-dernière ligne la dernière ligne autant de x possible pour que le RHS (terme droite) soit positif et le + petit possible (i.e. division euclidienne).

E.g. $\frac{11}{3} \approx 3$.... donc on créé une nouvelle ligne (a) $- 3 \times$ (b) :

$3 \times (-3) + 11 \times (1) = 2$ (c) = (a) $- 3 \times$ (b)

$0 - 3 \cdot 1$; $1 - 3 \times 0$; $11 - 3 \times 3$

$3 \times (4) + 11 \times (-1) = 1$ (d) = (b) − (c)

$1 - (-3)$; $0 - 1$; $3 - 2$

Pré-requis calculatoires

Autre méthode (plusieurs autres solutions) : on souhaite trouver u, v tels que $3u + 11v = 1$. Au début, on sait que:

$3 \times (0) + 11 \times (1) = 11$ (a)

$3 \times (1) + 11 \times (0) = 3$ (b)

Fixe!

On va alors faire des combinaisons linéaires de ces équations pour avoir $3 \times (?) + 11 \times (?) = 1$ ⇒ on soustrait à l'avant-dernière ligne la dernière ligne autant de x possible pour que le RHS (terme droite) soit positif et le + petit possible (i.e. division euclidienne).

E.g. $\frac{11}{3} \approx 3$.... donc on créé une nouvelle ligne (a) $- 3 \times$ (b) :

$3 \times (-3) + 11 \times (1) = 2$ (c) = (a) $- 3 \times$ (b)

$0 - 3.1$ $\quad 1 - 3 \times 0$ $\quad 11 - 3 \times 3$

$3 \times (4) + 11 \times (-1) = 1$ (d) = (b) - (c)

$1 - (-3)$ $\quad 0 - 1$ $\quad 3 - 2$

inverse!

$= 1$: gagné!

Pré-requis calculatoires

Autre méthode (plusieurs autres solutions) : on souhaite trouver u, v tels que $3u + 11v = 1$. Au début, on sait que:

$3 \times (0) + 11 \times (1) = 11$ (a)

$3 \times (1) + 11 \times (0) = 3$ (b)

flip!

On va alors faire des combinaisons linéaires de ces équations pour avoir $3 \times (?) + 11 \times (?) = 1$ ⇒ on soustrait à l'avant-dernière ligne la dernière ligne autant de x possible pour que le RHS (terme droite) soit positif et le + petit possible (i.e. division euclidienne).

E.g. $\frac{11}{3} \approx 3$.... donc on créé une nouvelle ligne (a) − 3 × (b) :

$3 \times (-3) + 11 \times (1) = 2$ (c) = (a) − 3 × (b)

0 − 3.1 ; 1 − 3×0 ; 11 − 3×3

$3 \times (4) + 11 \times (-1) = 1$ (d) = (b) − (c)

inverse! ; 1 − (−3) ; 0 − 1 ; 3 − 2

= 1 : gagné!

Pré-requis calculatoires

$23^{-1} \bmod 123 = ?$

Pré-requis calculatoires

$23^{-1} \bmod 123 = 47$

?

r													=	u	×	a	+	v	×	b
120													=	1	×	120	+	0	×	23
23													=	0	×	120	+	1	×	23
5	=	120	−	5	×	23							=	1	×	120	+	−5	×	23
3	=	23	−	4	×	5	=	1×23	−	4	×	(1×120 - 5×23)	=	−4	×	120	+	21	×	23
2	=	5	−	1	×	3	=	(1×120 − 5×23)	−	1	×	(−4×120 + 21×23)	=	5	×	120	+	-26	×	23
1	=	3	−	1	×	2	=	(−4×120 + 21×23)	−	1	×	(5×120 - 26×23)	=	−9	×	120	+	47	×	23

Vérification : $23 \times 47 \bmod 120 = 1$

Vers RSA

?

Alice souhaite définir le chiffrement $\mathrm{Enc}_{(e,n)}(m) \coloneqq me \bmod n$, où (e, n) définit la clé publique. Est-ce sécurisé ?

A. Oui

B. Non

$\Rightarrow$ Pour notre chiffrement, on souhaite utiliser cette même idée, mais avec un **inverse difficile à calculer** !

Vers RSA

?

Alice souhaite définir le chiffrement $\mathrm{Enc}_{(e,n)}(m) \coloneqq me \bmod n$, où (e, n) définit la clé publique. Est-ce sécurisé ?

A. Oui

B. Non ✓ Simple à casser : étant donné $c \coloneqq me$, à partir de e et n (publics), on sait calculer avec Euclide Étendu e^{-1}, et on déchiffre avec $ce^{-1} = mee^{-1} = m$.

$\Rightarrow$ Pour notre chiffrement, on souhaite utiliser cette même idée, mais avec un **inverse difficile à calculer** !

Vers RSA

?

Au lieu de définir le chiffré comme $me \bmod n$, pourquoi pas $m^e \bmod n$?

Vers RSA

Au lieu de définir le chiffré comme $me \bmod n$, pourquoi pas $m^e \bmod n$?

- $\Rightarrow$ Est-ce **possible/facile** à calculer ?

Vers RSA

Au lieu de définir le chiffré comme $me \bmod n$, pourquoi pas $m^e \bmod n$?

- $\Rightarrow$ Est-ce **possible/facile** à calculer ? Oui: **exponentiation rapide**: pour calculer m^e:
  - 1er cas, $e = 2x$ est pair : on calcule (récursivement) m^x, puis on calcule le carré $m^e = (m^x)^2$

Vers RSA

Au lieu de définir le chiffré comme $me \bmod n$, pourquoi pas $m^e \bmod n$?

- $\Rightarrow$ Est-ce **possible/facile** à calculer ? Oui: **exponentiation rapide**: pour calculer m^e:
  - 1er cas, $e = 2x$ est pair : on calcule (récursivement) m^x, puis on calcule le carré $m^e = (m^x)^2$
  - 2er cas, $e = 2x + 1$ est impair : on calcule (récursivement) m^x, puis on calcule $m^e =$ **quoi ?**

Vers RSA

? Au lieu de définir le chiffré comme $me \bmod n$, pourquoi pas $m^e \bmod n$?

- $\Rightarrow$ Est-ce **possible/facile** à calculer ? Oui: **exponentiation rapide**: pour calculer m^e:
  - 1er cas, $e = 2x$ est pair : on calcule (récursivement) m^x, puis on calcule le carré $m^e = (m^x)^2$
  - 2er cas, $e = 2x + 1$ est impair : on calcule (récursivement) m^x, puis on calcule $m^e = (m^x)^2 m$

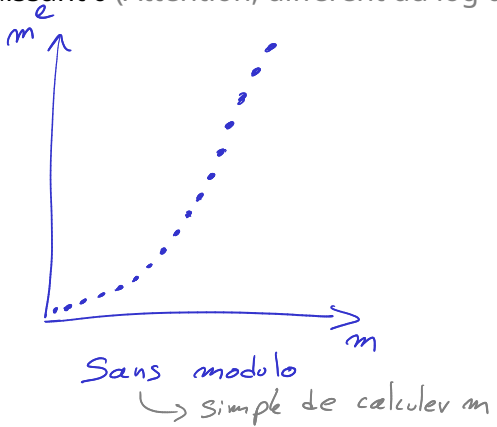
Vers RSA

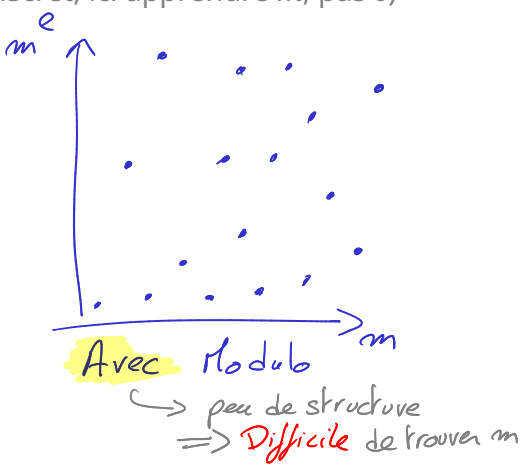
? Au lieu de définir le chiffré comme $me \bmod n$, pourquoi pas $m^e \bmod n$?

- $\Rightarrow$ Est-ce **possible/facile** à calculer ? Oui: **exponentiation rapide**: pour calculer m^e:
  - 1er cas, $e = 2x$ est pair : on calcule (récursivement) m^x, puis on calcule le carré $m^e = (m^x)^2$
  - 2er cas, $e = 2x + 1$ est impair : on calcule (récursivement) m^x, puis on calcule $m^e = (m^x)^2 m$
- $\Rightarrow$ Est-ce **possible/facile** de déchiffrer ? Dans quel cas ?

Vers RSA

Dans le cas général : **difficile** de calculer m à partir de $m^e \bmod n$, même en connaissant e (Attention, différent du log discret, ici apprendre m, pas e)





Vers RSA: retrouver m à partir de m^e

Si difficile de trouver m à partir de $m^e \bmod n$, comment déchiffrer ?

$\Rightarrow$ **Astuce** : si on connaît la **factorisation de** n, alors c'est **facile** !

Vers RSA: retrouver m à partir de m^e

Comment retrouver m à partir de m^e et la factorisation de n ?

1. Calculer l'**indicatrice d'Euler** $\phi(n)$ = nombre de nombres $\leq n$ premiers avec n. Si $n = pq$ avec p et q premiers, $\phi(n) = (p-1)(q-1)$.

Combien faut $\phi(10)$? Et $\phi(8)$?

Vers RSA: retrouver m à partir de m^e

Comment retrouver m à partir de m^e et la factorisation de n ?

1. Calculer l'**indicatrice d'Euler** $\phi(n)$ = nombre de nombres $\leq n$ premiers avec n. Si $n = pq$ avec p et q premiers, $\phi(n) = (p-1)(q-1)$.

Combien faut $\phi(10)$? Et $\phi(8)$?

- $10 = 2 \times 5$, 2 et 5 sont premiers, donc $\phi(10) = (2-1)(5-1) = 8$
- $8 = 2 \times 2 \times 2 \ldots$ Pas de la forme $n = pq$. Soit on utilise la formule générique $\phi(n) = \prod_i (p_i - 1) p_i^{k_i - 1}$ (k_i = multiplicité du facteur premier p_i), soit ici, petit exemple, on regarde tous les éléments $1, \ldots, 8$ uns par uns et on regarde ceux qui sont premier avec 8, i.e. $|\{1, 3, 5, 7\}| = 4$.

Vers RSA: retrouver m à partir de m^e

Comment retrouver m à partir de m^e et la factorisation de n ?

1. Calculer l'**indicatrice d'Euler** $\phi(n)$ = nombre de nombres $\leq n$ premiers avec n. Si $n = pq$ avec p et q premiers, $\phi(n) = (p-1)(q-1)$.
2. Calculer l'inverse de e modulo $\phi(n)$ (Euclide étendu), noté d: $de = 1 \mod \phi(n)$, i.e. $\exists k, de = 1 + k\phi(n)$.

Vers RSA: retrouver m à partir de m^e

Comment retrouver m à partir de m^e et la factorisation de n ?

1. Calculer l'**indicatrice d'Euler** $\phi(n)$ = nombre de nombres $\leq n$ premiers avec n. Si $n = pq$ avec p et q premiers, $\phi(n) = (p-1)(q-1)$.
2. Calculer l'inverse de e modulo $\phi(n)$ (Euclide étendu), noté d: $de = 1 \bmod \phi(n)$, i.e. $\exists k, de = 1 + k\phi(n)$.
3. Calculer $(m^e)^d$ et utiliser le théorème d'Euler pour conclure (cas particulier théorème de Lagrange, ordre d'un groupe divise le nombre d'éléments du groupe):

Théorème d'Euler

Si $x \in \mathbb{Z}_n^*$, alors $x^{\phi(n)} = 1 \bmod n$.

Vers RSA: retrouver m à partir de m^e

Comment retrouver m à partir de m^e et la factorisation de n ?

1. Calculer l'**indicatrice d'Euler** $\phi(n)$ = nombre de nombres $\leq n$ premiers avec n. Si $n = pq$ avec p et q premiers, $\phi(n) = (p-1)(q-1)$.
2. Calculer l'inverse de e modulo $\phi(n)$ (Euclide étendu), noté d: $de = 1 \mod \phi(n)$, i.e. $\exists k, de = 1 + k\phi(n)$.
3. Calculer $(m^e)^d$ et utiliser le théorème d'Euler pour conclure (cas particulier théorème de Lagrange, ordre d'un groupe divise le nombre d'éléments du groupe):

Théorème d'Euler

Si $x \in \mathbb{Z}_n^*$, alors $x^{\phi(n)} = 1 \mod n$.

Exercice : Vérifier avec python, avec $n = 19 \times 23$

Vers RSA: retrouver m à partir de m^e

Comment retrouver m à partir de m^e et la factorisation de n ?

1. Calculer l'**indicatrice d'Euler** $\phi(n)$ = nombre de nombres $\leq n$ premiers avec n. Si $n = pq$ avec p et q premiers, $\phi(n) = (p-1)(q-1)$.
2. Calculer l'inverse de e modulo $\phi(n)$ (Euclide étendu), noté d: $de = 1 \mod \phi(n)$, i.e. $\exists k, de = 1 + k\phi(n)$.
3. Calculer $(m^e)^d$ et utiliser le théorème d'Euler pour conclure (cas particulier théorème de Lagrange, ordre d'un groupe divise le nombre d'éléments du groupe):

Théorème d'Euler

Si $x \in \mathbb{Z}_n^*$, alors $x^{\phi(n)} = 1 \mod n$.

$\Rightarrow$ Combien vaut $(m^e)^d$?

Vers RSA: retrouver m à partir de m^e

Comment retrouver m à partir de m^e et la factorisation de n ?

1. Calculer l'**indicatrice d'Euler** $\phi(n)$ = nombre de nombres $\leq n$ premiers avec n. Si $n = pq$ avec p et q premiers, $\phi(n) = (p-1)(q-1)$.
2. Calculer l'inverse de e modulo $\phi(n)$ (Euclide étendu), noté d: $de = 1 \mod \phi(n)$, i.e. $\exists k, de = 1 + k\phi(n)$.
3. Calculer $(m^e)^d$ et utiliser le théorème d'Euler pour conclure (cas particulier théorème de Lagrange, ordre d'un groupe divise le nombre d'éléments du groupe):

Théorème d'Euler

Si $x \in \mathbb{Z}_n^*$, alors $x^{\phi(n)} = 1 \mod n$.

$\Rightarrow$ Combien vaut $(m^e)^d$? $(m^e)^d = m^{ed} = m^{1+k\phi(n)} = m(m^{\phi(n)})^k = m1^k = m$ Gagné !

RSA

Fonction RSA

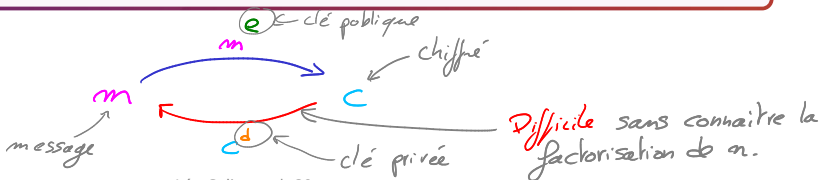
On a maintenant tous les éléments pour définir la fonction RSA :

Définition (fonction RSA)

Étant donné $n = pq$, et e (clé publique) et d (clé privée) tels que $ed = 1 \bmod \phi(n)$, on définit la fonction RSA.

$$f(m) := m^e \bmod n$$

On a alors $f^{-1}(c) := c^d \bmod n$. d = **porte dérobée** (trapdoor).



Quelle garantie sur la difficulté de RSA ?

- Meilleur attaque connue = factoriser (et donc récupérer $\phi(n)$ et d).
  - Algo naîf pour factoriser n = $O(\sqrt{n})$ opérations.
    Est-ce polynomial ?

Quelle garantie sur la difficulté de RSA ?

- Meilleur attaque connue = factoriser (et donc récupérer $\phi(n)$ et d).
  - Algo naîf pour factoriser n = $O(\sqrt{n})$ opérations.
    Est-ce polynomial ?  pas polynomial en la taille du nombre (cf. premier cours)

Quelle garantie sur la difficulté de RSA ?

- Meilleur attaque connue = factoriser (et donc récupérer $\phi(n)$ et d).
  - Algo naîf pour factoriser n = $O(\sqrt{n})$ opérations.
    Est-ce polynomial ?  pas polynomial en la taille du nombre (cf. premier cours)
  - Meilleurs algo de factorisation classique = Generalized Number Field Sieve (GNFS) = $O(n^{(n/\log n)^{1/3}})$ = **pas efficace !** (super-polynomial)

Quelle garantie sur la difficulté de RSA ?

- Meilleur attaque connue = factoriser (et donc récupérer $\phi(n)$ et d).
  - Algo naïf pour factoriser n = $O(\sqrt{n})$ opérations.
    Est-ce polynomial ? ✗ pas polynomial en la taille du nombre (cf. premier cours)
  - Meilleurs algo de factorisation classique = Generalized Number Field Sieve (GNFS) = $O(n^{(n/\log n)^{1/3}})$ = **pas efficace !** (super-polynomial)
  - Mais avec ordinateur quantique, algo de **Shor** = factorise en $\approx \tilde{O}(\log^3 N)$ = polynomial !! (Regev améliore à $O(\log^{2.5} N)$)
    $=>$ **RSA pas sécurisé contre ordinateurs quantiques**
    (pour l'instant trop petits pour poser problèmes, mais attention "**store now decrypt later**"!)

Et **en pratique**, quelle taille ?

- Plus gros challenge RSA cassé = RSA-250 (829 bits)
- $\Rightarrow$ On **recommande 1024 ou 4096 bits** (pour p et q)

RSA dans la vraie vie

RSA dans la vraie vie

RSA (original, NE PAS UTILISER)

On a envie de définir le chiffrement naturel (historique) suivant:

$\underline{\mathsf{Gen}(1^\lambda)}$:
- $p, q \xleftarrow{\$}$ entiers premiers de taille λ
- $n \coloneqq pq$
- $e \xleftarrow{\$} \{1, \dots, \phi(n)\}$ tel que $\mathsf{pgcd}(e, \phi(n)) = 1$
- $d \coloneqq e^{-1} \bmod \phi(n)$
- $p_k \coloneqq (N, e)$
- $s_k \coloneqq (N, d)$
- **return** (p_k, s_k)

Test primalité $O(\log^3 N)$, souvent Miller-Rabin (probabiliste)

$\underline{\mathsf{Enc}((N, e), m)}$:
return $m^e \bmod N$

$\underline{\mathsf{Dec}((N, d), c)}$:
return $c^d \bmod N$

RSA dans la vraie vie

Est-ce que RSA originel est IND-CPA sécurisé ?

A. Oui

B. Non

RSA dans la vraie vie

Est-ce que RSA originel est IND-CPA sécurisé ?

A. Oui

B. Non ✓

- C'est un chiffrement **déterministe**, ne **peut pas être sécurisé** !
- Attaque quadratiquement meilleure que brute-force existe !
- + beaucoup de détails : e.g. si e et m petits, alors le modulo ne s'applique pas = facile à casser…

Détails : *Introduction to modern cryptography*, Katz and Lindell.

RSA dans la vraie vie

Padded RSA (NE PAS UTILISER)

Solution pour rendre RSA non-déterministe : première idée = ajouter padding aléatoire à m lors du chiffrement :

$\underline{\text{Enc}((N,e), m \in \{0,1\}^l):}$
$r \xleftarrow{\$} \{0,1\}^{|n|-l}$ jusqu'à ce que $r\|m \in \mathbb{Z}_n^*$
return $(r\|m)^e \bmod N$

$\underline{\text{Dec}((N,d), c):}$
return l derniers bits de $c^d \bmod N$

RSA dans la vraie vie

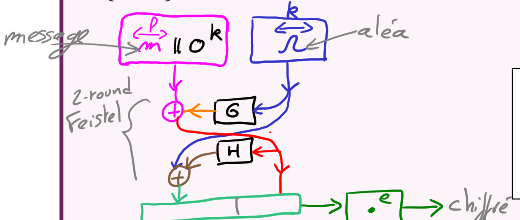
Est-ce vraiment sécurisé ?

- Trop **petit padding : non sécurisé**
- Très grand padding ($|m| = 1$) : on a une **preuve de sécurité** !
  Idée: le plus petit bit du message est un hard-core predicate : utiliser ce bit pour stoquer m, reste complètement aléatoire. **Problème : très inefficace !!** (intérêt théorique, KEM un peu plus efficace, pas suffisant)
- Padding intermédiaire : **on ne sait pas** trop, pas de preuve, pas d'attaques
- **Non CCA-sécurisé** !
  - Padded RSA (plus précisément une variante PKCS#1) est malléable, donc vulnérable.
  - PKCS#1 rajoute des conditions limitant maléabilité… mais **cassé** avec attaque type padding oracle attaque par Bleichenbacher (1998), et en pratique sur SSL !

Dans la vraie vie : **combine RSA avec fonction de hash = RSA-OAEP**

Définition (RSA-OAEP)

Soit l (taille m), k tels que $k = \Theta(n)$ et $l + 2k < |n|$. Le chiffrement RSA-OAEP est défini comme suit, avec $G\colon \{0,1\}^k \to \{0,1\}^{l+k}$ et $H\colon \{0,1\}^{l+k} \to \{0,1\}^k$ deux fonctions de hash :



$\underline{\mathsf{Enc}((N,e), m \in \{0,1\}^l)\text{:}}$
$r \leftarrow\!\!\$\ \{0,1\}^r$
return $((m\|0^k) \oplus G(r)\|r \oplus H(t))^e \bmod N$

(la génération de clé est toujours la même)

RSA-OAEP

?

Exercice : quel est l'algorithme de déchiffrement ? Dans quel cas donner une erreur de déchiffrement ?

?

Exercice : quel est l'algorithme de déchiffrement ? Dans quel cas donner une erreur de déchiffrement ?
Déchiffrement évident. Pour l'erreur, deux cas:

- si $|c^d| > l + 2k$
- si on retrouve pas les 0^k après m

RSA-OAEP

Théorème (sécurité RSA-OAEP)

Dans le modèle de l'oracle aléatoire (G et H = vrai aléa), en en supposant que la fonction RSA est one-way, RSA-OAEP est CCA sécurisé.

RSA-OAEP

Théorème (sécurité RSA-OAEP)

Dans le modèle de l'oracle aléatoire (G et H = vrai aléa), en en supposant que la fonction RSA est one-way, RSA-OAEP est CCA sécurisé.

Standardisé **PKCS#1v2.0**.

RSA-OAEP

Théorème (sécurité RSA-OAEP)

Dans le modèle de l'oracle aléatoire (G et H = vrai aléa), en en supposant que la fonction RSA est one-way, RSA-OAEP est CCA sécurisé.

Standardisé **PKCS#1v2.0**.

Malgré cette preuve, une attaque CCA en pratique ! (Manger, 2001) **Comment est-ce possible** avec la preuve de sécurité ?

RSA-OAEP

Théorème (sécurité RSA-OAEP)

Dans le modèle de l'oracle aléatoire (G et H = vrai aléa), en en supposant que la fonction RSA est one-way, RSA-OAEP est CCA sécurisé.

Standardisé **PKCS#1v2.0**.

Malgré cette preuve, une attaque CCA en pratique ! (Manger, 2001) **Comment est-ce possible** avec la preuve de sécurité ?

$\Rightarrow$ Dans l'algorithme/modèle, deux source d'erreur, mais retourne la même erreur. En pratique, l'implémentation renvoyait deux erreurs **différentes**. **Suffisant pour casser la sécurité !**

Détails additionnels

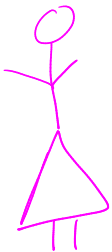
Attention, encore de **nombreux détails** :

- Il faut choisir les **paramètres RSA correctement sinon attaques**, entre autre:
  - Grands p et q sinon attaque rho de Pollard
  - $|p - q|$ doit être grand sinon brute force possible
  - $p - 1$ et $q - 1$ non friables (attaque $p - 1$ de Pollard)
  - $p + 1$ et $q + 1$ doivent avoir de grands facteurs premiers (attaque Williams)
  - ne jamais ré-utiliser les clés privées
  - e doit être grand (attaque Håstad)
  - d doit être supérieur à $\frac{1}{3}n^{\frac{1}{4}}$ (attaque Wiener)
  - …
- On peut calculer m^d plus **efficacement** en connaissant p et q via le Chinease Reminder Theorem (en calculant deux plus petites exponentiations et en les combinant).

Signature avec RSA

Signature

I. Comment prouver qu'on connait la clé sans la révéler



Signature

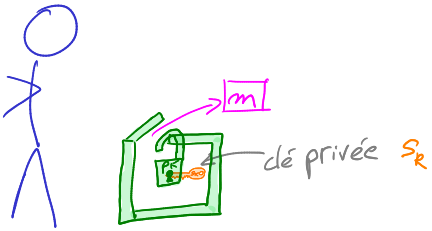
I. Comment prouver qu'on connait la clé sans la révéler

1. Alice envoit un chiffré

Signature

I. Comment prouver qu'on connait la clé sans la reveler

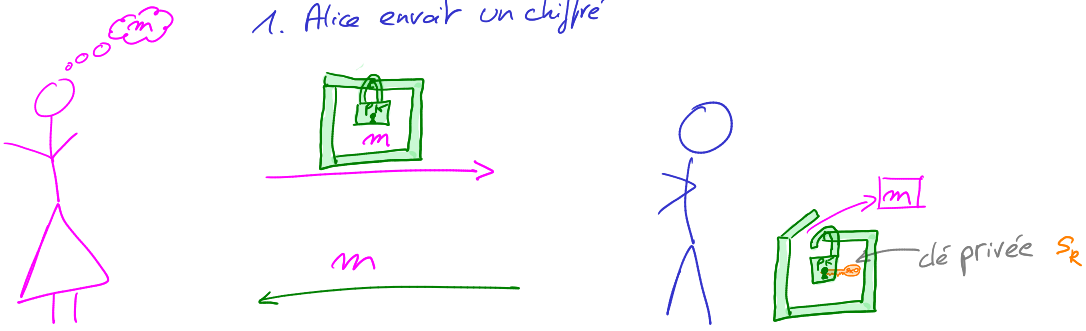
1. Alice envoit un chiffré



Signature

I. Comment prouver qu'on connait la clé sans la révéler

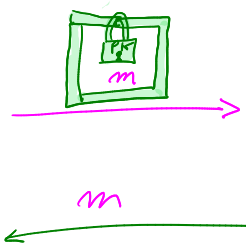
1. Alice envoit un chiffré

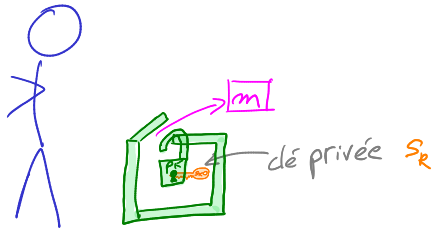


Signature

I. Comment prouver qu'on connait la clé sans la révéler

1. Alice envoit un chiffré



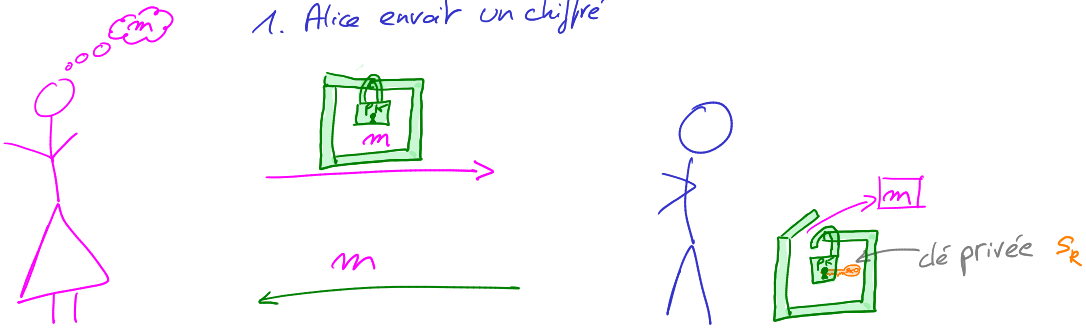


2. Si Bob arrive à le déchiffrer, il a la clé privée ! ≈ signature

Signature

I. Comment prouver qu'on connait la clé sans la révéler

1. Alice envoit un chiffré

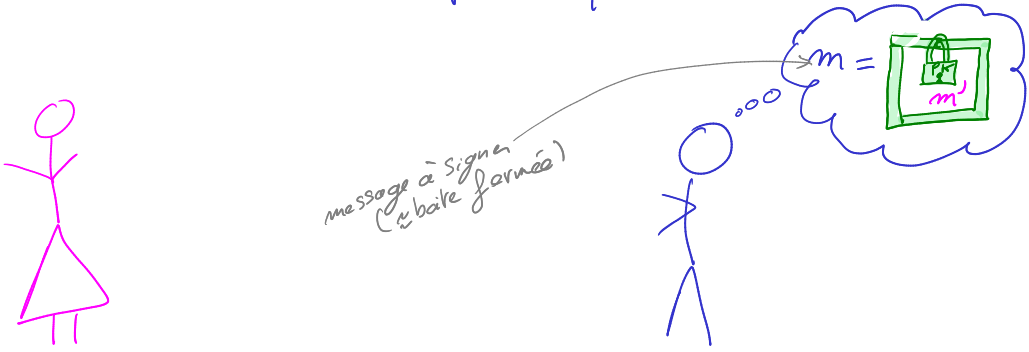


2. Si Bob arrive à le déchiffrer, il a la clé privée ! ≈ signature

Pb : Comment lier la signature à un message ?

Signature

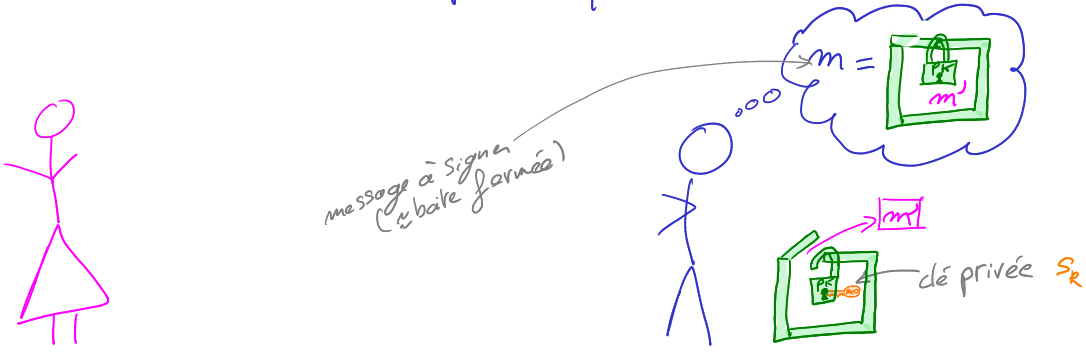
I. Comment prouver qu'on connait la clé sans la révéler



Pb : Comment lier la signature à un message ?

Signature

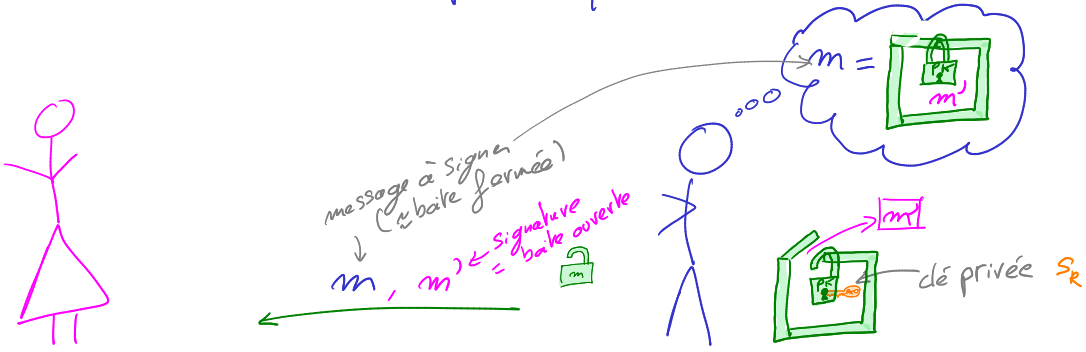
I. Comment prouver qu'on connait la clé sans la révéler



Pb : Comment lier la signature à un message ?

Signature

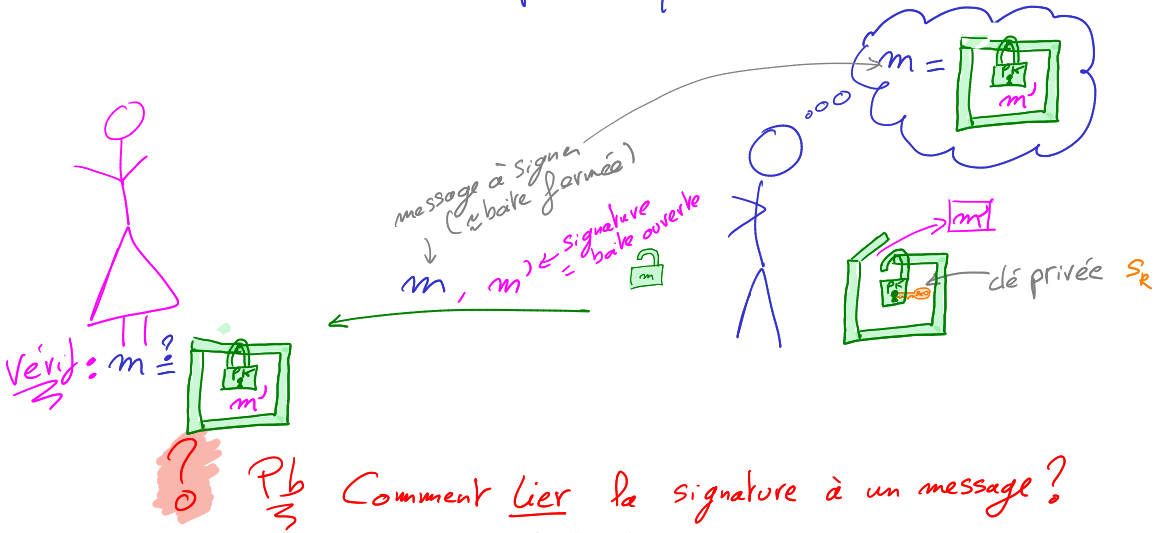
I. Comment prouver qu'on connait la clé sans la révéler



Pb : Comment lier la signature à un message ?

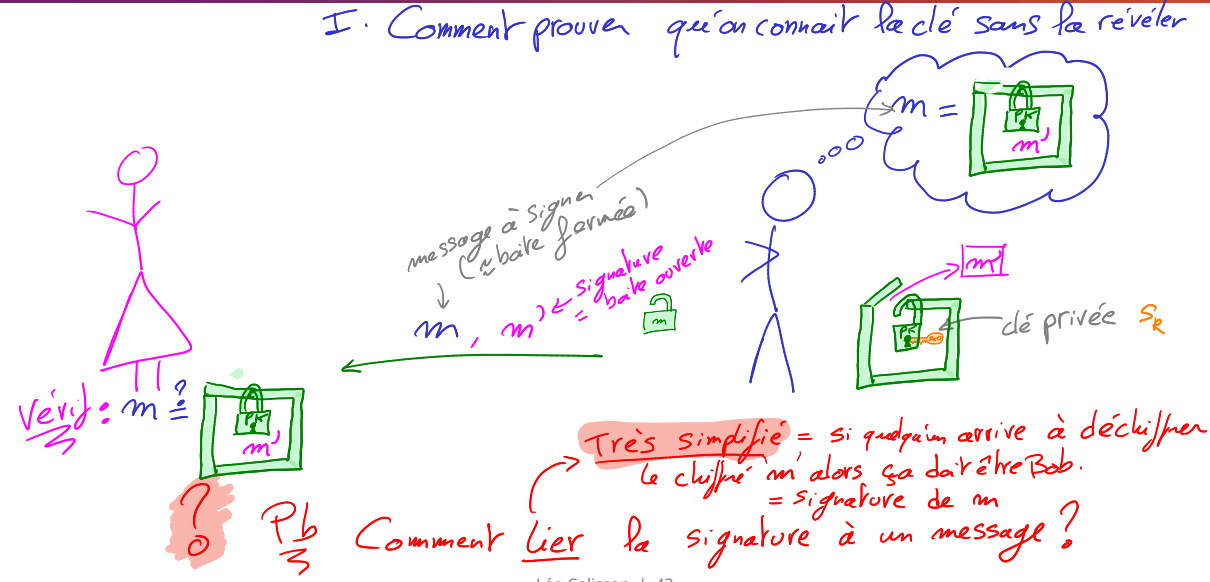
Signature

I. Comment prouver qu'on connait la clé sans la révéler



Pb Comment lier la signature à un message ?

Signature


I. Comment prouver qu'on connait la clé sans la révéler
m =
m'
message à signer
(= boîte fermée)
m , m'
= signature
boîte ouverte
m
m'
clé privée S_R
Vérif : m ?=
m'
Très simplifié = si quelqu'un arrive à déchiffrer le chiffré m' alors ça doit être Bob.
= signature de m
Pb
Comment lier la signature à un message ?

Signature RSA textbook

Signature = une personne peut signer (avec clé **privée** donc !), tout le monde peut vérifier (avec clé **publique**)

Idée : si on prouve qu'on arrive à "déchiffrer" n'importe quel chiffré, c'est qu'on connait la clé secrète !

Définition (Textbook RSA, ne pas utiliser !)

On définit donc cette première version (non sécurisée) de signature RSA dite "textbook" :

$\underline{\mathsf{Sign}((N,d),m)\text{:}}$
return $m^d \bmod N$

$\underline{\mathsf{Verif}((N,e),m,\sigma)\text{:}}$
return $\sigma^e \bmod N \stackrel{?}{=} m$

Formaliser la sécurité

Définition (sécurité signature)

De manière analogue aux MACs, on définit la sécurité d'une signature par:

$\mathcal{L}_{\text{sig-real}}$

$(p_k, s_k) \leftarrow \text{Gen}(1^\lambda)$

GETPK$()$:

  return p_k

GETSIG(m):

  return $\text{Sign}(s_k, m)$

VERSIG(m, σ):

  return $\text{Verif}(s_k, m, \sigma)$

Signature RSA textbook

Trouvez deux attaques contre la signature Textbook RSA :

- une forge existentielle sans aucun appel à l'oracle de signature
- une forge universelle avec appel à l'oracle

TODO: exercice **caseine**

Signature RSA textbook

Trouvez deux attaques contre la signature Textbook RSA :

- une forge existentielle sans aucun appel à l'oracle de signature
- une forge universelle avec appel à l'oracle

?

TODO: exercice **caseine**
Plusieurs attaques :

- existentielle : on prend σ au hasard, alors σ^e est une signature de σ !
- existentielle (plus puissante, 1 appel) : on signe m, pour obtenir σ, alors σ^2 est une signature pour m^2 !
- universelle (2 appels) : pour avoir une signature de m = décompose $m = m_1 m_2$, on signe m_1 et m_2 avec oracle (=σ_1, σ_2), alors $\sigma_1 \sigma_2 = m_1^d m_2^d = (m_1 m_2)^d = m^d$ est une signature de m !

RSA-FDH (Full Domain Hash)

But : casser la relation entre signature et message. On définit donc cette version améliorée de signature RSA, basée sur une fonction de hash :

Définition (RSA-FDH)

Soit H une fonction de hash avec pour sortie $\mathbb{Z}_N^*$[a]. On définit :

$\mathsf{Sign}((N,d),m)$:
return $H(m)^d \bmod N$

$\mathsf{Verif}((N,e),m,\sigma)$:
$y \coloneqq \sigma^e \bmod N \stackrel{?}{=} m$
return $H(m) \stackrel{?}{=} y$

(Gen comme d'habitude)

RSA-FDH

Prouver que cette signature est correcte.

RSA-FDH

? Si h n'est pas résistance aux collisions, peut-on casser la signature ?

?

Si h n'est pas résistance aux collisions, peut-on casser la signature ?
Oui, si $H(m) = H(m')$, alors en signant m, on a aussi une signature pour m'.

RSA-FDH

Théorème (sécurité RSA-FDH)

Si H est modélisé comme un oracle aléatoire et si le problème RSA[a] est difficile (pour les paramètres de la génération de clé), alors **RSA-FDH est une signature sécurisée**.

[a] I.e. calculer $y^{1/e} \bmod N$.

$\Rightarrow$ RSA-FDH est à la base de la **norme** RSA PKCS#1 v1.5 et v2.

Conclusion

Conclusion

- Cryptographie clé publique = communiquer **sans se rencontrer** au préalable
- **Moins efficace** que clé privée = typiquement on échange des clés via crypto asymétrique, puis on utilise crypto symétrique (= chiffrement hybride)
- **RSA** = crypto clé publique très commune basée sur la **difficulté de factoriser** :
  - chiffrement = RSA-OAEP
  - signature = RSA-FDH/RSA PKCS#1
- De nombreuses **attaques possibles si on ne fait pas très attention**. En particulier

NE PAS UTILISER RSA “TEXTBOOK” !!!

- Connaître ses règles de calcul de puissances… c’est important !