Cryptographie à clé publique. Principe de Kerckhoffs

Cryptographie à clé publique.
Principe de Kerckhoffs

> *La sécurité d'un système de chiffrement ne doit reposer que sur le secret de la clé.*

Cryptographie à clé publique.
Principe de Kerckhoffs

La sécurité d'un système de chiffrement ne doit reposer que sur le secret de la clé.

L'ennemi peut avoir connaissance du système de chiffrement.

Cryptographie à clé publique.
Principe de Kerckhoffs

La sécurité d'un système de chiffrement ne doit reposer que sur le secret de la clé.

L'ennemi peut avoir connaissance du système de chiffrement.

- Contraire à l'intuition qui est de dissimuler le maximum de choses possibles

Cryptographie à clé publique.
Principe de Kerckhoffs

La sécurité d'un système de chiffrement ne doit reposer que sur le secret de la clé.

L'ennemi peut avoir connaissance du système de chiffrement.

- Contraire à l'intuition qui est de dissimuler le maximum de choses possibles
- Un mécanisme connu de tous sera testé, attaqué, étudié, et utilisé s'il est robuste

Cryptographie à clé publique.
Principe de Kerckhoffs

La sécurité d'un système de chiffrement ne doit reposer que sur le secret de la clé.

L'ennemi peut avoir connaissance du système de chiffrement.

- Contraire à l'intuition qui est de dissimuler le maximum de choses possibles
- Un mécanisme connu de tous sera testé, attaqué, étudié, et utilisé s'il est robuste
- Seule la clé utilisée reste secrète

Complexité de la factorisation

- $5 \times 7 = ?$
- $35 = ?$

Complexité de la factorisation

- $5 \times 7 = ?$
- $35 = ?$
- Factoriser 1591
- Calculer 37×43

Complexité de la factorisation

- $5 \times 7 = ?$
- $35 = ?$
- Factoriser 1591
- Calculer 37×43
- Calculer $p \times q$ est plus facile que de factoriser $n = pq$

Complexité de la factorisation

- $5 \times 7 = ?$
- $35 = ?$
- Factoriser 1591
- Calculer 37×43
- Calculer $p \times q$ est plus facile que de factoriser $n = pq$

- La ***complexité*** estime le temps de calcul (ou le nombre d'opérations élémentaires) nécessaire pour effectuer une opération

Complexité de la factorisation

- $5 \times 7 =?$
- $35 =?$
- Factoriser 1591
- Calculer 37×43
- Calculer $p \times q$ est plus facile que de factoriser $n = pq$

- La ***complexité*** estime le temps de calcul (ou le nombre d'opérations élémentaires) nécessaire pour effectuer une opération
- **Addition**
  - La somme de deux chiffres (par exemple $6 + 8$) est de complexité 1
  - La somme de deux entiers à n chiffres est de complexité n
  - Ex. $1234 + 2323$: 4 additions de chiffres

Complexité de la factorisation

- $5 \times 7 = ?$
- $35 = ?$
- Factoriser 1591
- Calculer 37×43
- Calculer $p \times q$ est plus facile que de factoriser $n = pq$

- La ***complexité*** estime le temps de calcul (ou le nombre d'opérations élémentaires) nécessaire pour effectuer une opération
- **Addition**
  - La somme de deux chiffres (par exemple $6 + 8$) est de complexité 1
  - La somme de deux entiers à n chiffres est de complexité n
  - Ex. $1234 + 2323$: 4 additions de chiffres
- **Multiplication**
  - La multiplication de deux entiers à n chiffres est de complexité n^2
  - Ex. 1234×2323 : 16 multiplications de chiffres

Complexité de la factorisation

- $5 \times 7 = ?$
- $35 = ?$
- Factoriser 1591
- Calculer 37×43
- Calculer $p \times q$ est plus facile que de factoriser $n = pq$

- La ***complexité*** estime le temps de calcul (ou le nombre d'opérations élémentaires) nécessaire pour effectuer une opération
- **Addition**
  - La somme de deux chiffres (par exemple $6 + 8$) est de complexité 1
  - La somme de deux entiers à n chiffres est de complexité n
  - Ex. $1234 + 2323$: 4 additions de chiffres
- **Multiplication**
  - La multiplication de deux entiers à n chiffres est de complexité n^2
  - Ex. 1234×2323 : 16 multiplications de chiffres
- **Factorisation** : $\exp(4n^{\frac{1}{3}})$

Multiplier et factoriser des nombres à n chiffres

n	multiplication	factorisation
3	9	320
4	16	572
5	25	934
10	100	5 528
50	2 500	2 510 835
100	10 000	115 681 968
200	40 000	14 423 748 780

Fonctions à sens unique

- Fonction f
- Connaissant x, calcul «facile» de $f(x)$
- Pour un y, trouver x tel que $y = f(x)$ est «difficile»
- Fonction à sens unique à ***trappe***

Exemple :

$$f : x \longmapsto x^3 \pmod{100}$$

Trouver x tel que $x^3 \equiv 11 \pmod{100}$

- Recherche exhaustive, tester 0, 1, 2, 3, ..., 99

$$71^3 = 357\,911 \equiv 11 \pmod{100}$$

Fonctions à sens unique

- Fonction f
- Connaissant x, calcul «facile» de $f(x)$
- Pour un y, trouver x tel que $y = f(x)$ est «difficile»
- Fonction à sens unique à ***trappe***

Exemple :

$$f : x \longmapsto x^3 \pmod{100}$$

Trouver x tel que $x^3 \equiv 11 \pmod{100}$

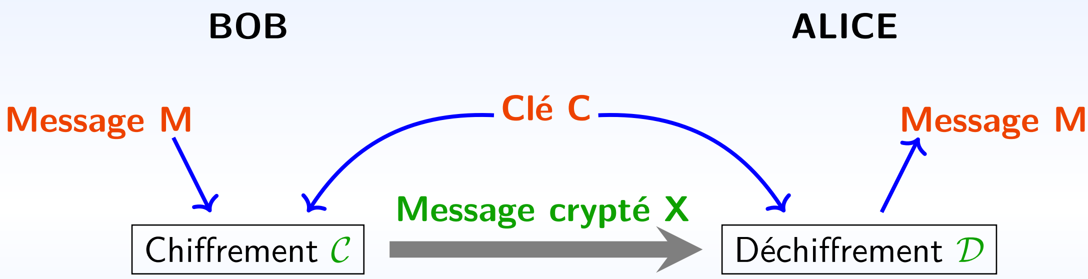
- Recherche exhaustive, tester 0, 1, 2, 3, ..., 99

$$71^3 = 357\,911 \equiv 11 \pmod{100}$$

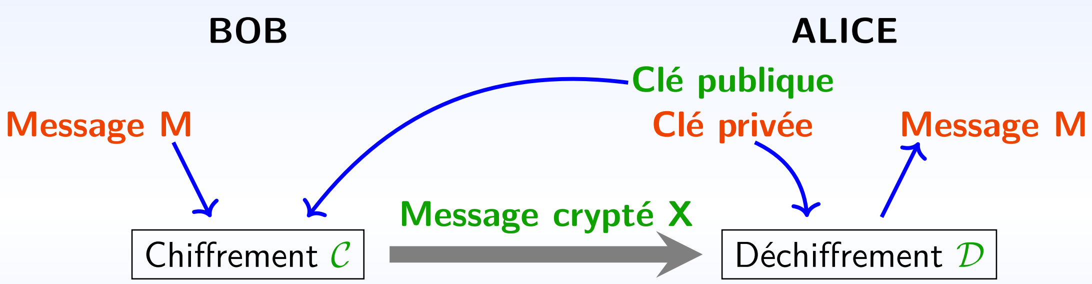
- Trappe secrète : $y \longmapsto y^7 \pmod{100}$ qui fournit directement le résultat !

$$11^7 = 19\,487\,171 \equiv 71 \pmod{100}$$

Chiffrement à clé privée



Chiffrement à clé publique



Paramètres d'un ***chiffrement à clé publique***

1. les fonctions de chiffrement et de déchiffrement $\mathcal{C}$ et $\mathcal{D}$
2. la clé publique du destinataire qui paramètre la fonction $\mathcal{C}$
3. la clé privée du destinataire qui paramètre la fonction $\mathcal{D}$

$$\text{Trouver } x \text{ tel que } x^3 \equiv 11 \pmod{100}$$

Paramètres d'un ***chiffrement à clé publique***

1. les fonctions de chiffrement et de déchiffrement $\mathcal{C}$ et $\mathcal{D}$
2. la clé publique du destinataire qui paramètre la fonction $\mathcal{C}$
3. la clé privée du destinataire qui paramètre la fonction $\mathcal{D}$

$$\text{Trouver } x \text{ tel que } x^3 \equiv 11 \pmod{100}$$

1. $\mathcal{C} : x \longmapsto x^{\mathbf{?}} \pmod{100}$ et $\mathcal{D} : x \longmapsto x^{\mathbf{?}} \pmod{100}$
2. la clé publique d'Alice **3** $\quad \mathcal{C} : x \longmapsto x^{\mathbf{3}} \pmod{100}$
3. la clé privée d'Alice **7** $\quad \mathcal{D} : x \longmapsto x^{\mathbf{7}} \pmod{100}$

Théorème Théorème d'Euler

Si n et a sont premiers entre eux, alors

$$\boxed{a^{\varphi(n)} \equiv 1 \pmod{n}}$$

Théorème Petit théorème de Fermat

Si p est un nombre premier et $a \in \mathbb{Z}$ alors

$$\boxed{a^p \equiv a \pmod{p}}$$

Théorème Théorème d'Euler

Si n et a sont premiers entre eux, alors

$$\boxed{a^{\varphi(n)} \equiv 1 \pmod{n}}$$

Théorème Petit théorème de Fermat

Si p est un nombre premier et $a \in \mathbb{Z}$ alors

$$\boxed{a^{p} \equiv a \pmod{p}}$$

Corollaire

Si p ne divise pas a alors

$$\boxed{a^{p-1} \equiv 1 \pmod{p}}$$

Théorème Petit théorème de Fermat amélioré

Soient p et q deux nombres premiers distincts et soit $n = pq$. Pour tout $a \in \mathbb{Z}$ tel que $\text{pgcd}(a, n) = 1$ alors :

$$\boxed{a^{(p-1)(q-1)} \equiv 1 \pmod{n}}$$

Théorème Petit théorème de Fermat amélioré

Soient p et q deux nombres premiers distincts et soit $n = pq$. Pour tout $a \in \mathbb{Z}$ tel que $\text{pgcd}(a, n) = 1$ alors :

$$\boxed{a^{(p-1)(q-1)} \equiv 1 \pmod{n}}$$

- On note que $\varphi(n) = (p-1)(q-1)$ (indicatrice d'Euler)
- $\text{pgcd}(a, n) = 1 \iff p$ et q ne divisent par a

Théorème Petit théorème de Fermat amélioré

Soient p et q deux nombres premiers distincts et soit $n = pq$. Pour tout $a \in \mathbb{Z}$ tel que $\mathrm{pgcd}(a, n) = 1$ alors :

$$\boxed{a^{(p-1)(q-1)} \equiv 1 \pmod{n}}$$

- On note que $\varphi(n) = (p-1)(q-1)$ (indicatrice d'Euler)
- $\mathrm{pgcd}(a, n) = 1 \iff p$ et q ne divisent par a
- Exemple : $p = 5,\ q = 7$
  - $n = p \times q = 35$
  - $(p-1) \times (q-1) = 4 \times 6 = 24$
  - Pour $a = 1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18, \ldots \quad a^{24} \equiv 1 \pmod{35}$

Principe de l'**algorithme d'Euclide**

$$\mathrm{pgcd}(a, b) = \mathrm{pgcd}(b, a \pmod b)$$

Principe de l'**algorithme d'Euclide**

$$\text{pgcd}(a, b) = \text{pgcd}(b, a \pmod{b})$$

$$a_0 = a, b_0 = b \qquad \text{puis} \qquad \begin{cases} a_{i+1} & = & b_i \\ b_{i+1} & = & a_i \pmod{b_i} \end{cases}$$

Principe de l'**algorithme d'Euclide**

$$\mathrm{pgcd}(a, b) = \mathrm{pgcd}(b, a \pmod{b})$$

$$a_0 = a, b_0 = b \qquad \text{puis} \qquad \begin{cases} a_{i+1} &= b_i \\ b_{i+1} &= a_i \pmod{b_i} \end{cases}$$

Principe de l'**algorithme d'Euclide**

$$\mathrm{pgcd}(a, b) = \mathrm{pgcd}(b, a \pmod b)$$

$$a_0 = a, b_0 = b \qquad \text{puis} \qquad \begin{cases} a_{i+1} &= b_i \\ b_{i+1} &= a_i \pmod{b_i} \end{cases}$$

Algorithme euclide.py (1)

```
fonction euclide(a,b): (a et b sont positifs)
    tant que b est non nul
        calculer le reste r de la DE de a par b
        a prend la valeur b
        b prend la valeur r
    renvoyer a
```

Principe de l'**algorithme d'Euclide**

$$\mathrm{pgcd}(a, b) = \mathrm{pgcd}(b, a \pmod b)$$

$$a_0 = a, b_0 = b \qquad \text{puis} \qquad \begin{cases} a_{i+1} & = & b_i \\ b_{i+1} & = & a_i \pmod{b_i} \end{cases}$$

Algorithme euclide.py (1)

```
def euclide(a,b):
    while b !=0 :
        a , b = b , a % b
    return a
```

L'***algorithme d'Euclide étendu*** pour obtenir les coefficients de Bézout u, v tels que $au + bv = \mathrm{pgcd}(a, b)$

L'***algorithme d'Euclide étendu*** pour obtenir les coefficients de Bézout u, v tels que $au + bv = \text{pgcd}(a, b)$

$$u_0 = 1 \qquad u_1 = 0 \qquad v_0 = 0 \qquad v_1 = 1$$

Puis pour $i \geqslant 1$

$$u_{i+1} = u_{i-1} - q_i u_i \qquad v_{i+1} = v_{i-1} - q_i v_i$$

où q_i est le quotient de la division euclidienne de a_i par b_i

L'***algorithme d'Euclide étendu*** pour obtenir les coefficients de Bézout u, v tels que $au + bv = \text{pgcd}(a, b)$

$$u_0 = 1 \qquad u_1 = 0 \qquad v_0 = 0 \qquad v_1 = 1$$

Puis pour $i \geqslant 1$

$$u_{i+1} = u_{i-1} - q_i u_i \qquad v_{i+1} = v_{i-1} - q_i v_i$$

où q_i est le quotient de la division euclidienne de a_i par b_i

Algorithme euclide.py (2)

```
def euclide_etendu(a,b):
    u = 1 ; uu = 0
    v = 0 ; vv = 1
    while b !=0 :
        q = a // b
        a , b = b , a % b
        uu , u = u - q*uu , uu
        vv , v = v - q*vv , vv
    return (a,u,v)
```

- Problème difficile :
  factoriser un entier produit de deux **nombres premiers distincts**
- Clé publique et clé secrète :
  calculées à l'aide de l'**algorithme d'Euclide** et des **coefficients de Bézout**
- Environnement :
  calculs **modulo** un entier
- Déchiffrement :
  fonctionne grâce au **petit théorème de Fermat**

Bob veut envoyer un message secret à Alice

1. Alice prépare une clé publique et une clé privée
2. Bob utilise la clé publique d'Alice pour crypter son message
3. Alice reçoit le message crypté et le déchiffre grâce à sa clé privée

Étape 1. Préparation des clés

Étape 1.1 Choix de deux nombres premiers

Informations privées, informations publiques.

Alice effectue les opérations suivantes

- choix de deux nombres premiers distincts p et q
- calcul de $n = p \times q$
- calcul de $\varphi(n) = (p-1) \times (q-1) \ = \varphi(n)$

Exemple facile

- $p = 5$ et $q = 17$
- $n = p \times q = 85$
- $\varphi(n) = (p-1) \times (q-1) = \varphi(n) = 64$

Étape 1. Préparation des clés
Étape 1.2 Choix d'un exposant et calcul de son inverse

- Alice choisit un exposant e tel que $\text{pgcd}(e, \varphi(n)) = 1$
- Alice calcule l'inverse d de e module $\varphi(n)$ par l'algorithme d'Euclide étendu : $d \times e \equiv 1 \pmod{\varphi(n)}$

Exemple facile

- $e = 5$ et on a bien $\text{pgcd}(e, \varphi(n)) = \text{pgcd}(5, 64) = 1$
-
  - $5 \times 13 + 64 \times (-1) = 1$
  - donc $5 \times 13 \equiv 1 \pmod{64}$
  - l'inverse de e modulo $\varphi(n)$ est $d = 13$

Étape 1. Préparation des clés

Étape 1.3 Clé publique

La ***clé publique*** d'Alice est constituée des deux nombres

$$\boxed{n \quad \text{et} \quad e}$$

Étape 1.4 Clé privée

Alice garde pour elle sa ***clé privée***

$$\boxed{d}$$

Exemple facile

$n = 85$ et $e = 5$

$d = 13$

Étape 2. Chiffrement du message
Étape 2.1 Message

- Bob veut envoyer un message secret à Alice
- Il transforme son message en un (ou plusieurs) entier m
- L'entier m vérifie $0 \leqslant m < n$

Exemple facile

$m = 10$

Étape 2. Chiffrement du message
Étape 2.2 Message chiffré

- Bob récupère la clé publique d'Alice : n et e
- Il calcule le message chiffré $\boxed{x \equiv m^e \pmod{n}}$
- Il transmet ce message x à Alice

Exemple facile

- $m = 10$, $n = 85$ et $e = 5$
- $x \equiv m^e \pmod{n} \equiv 10^5 \pmod{85}$
  - $10^2 = 100 \equiv 15 \pmod{85}$
  - $10^4 = (10^2)^2 \equiv 15^2 \equiv 225 \equiv 55 \pmod{85}$
  - $10^5 = 10^4 \times 10 \equiv 55 \times 10 \equiv 550 \equiv 40 \pmod{85}$
- Le message chiffré est donc $x = 40$

Étape 3. Déchiffrement du message

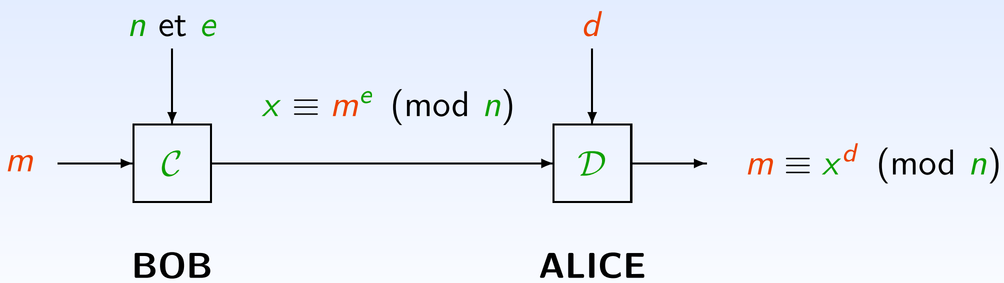
- Alice reçoit le message x chiffré par Bob
- Alice le décrypte à l'aide de sa clé privée d
- $\boxed{m \equiv x^d \pmod{n}}$

Exemple facile

- $x = 40$, $d = 13$, $n = 85$

  $40^{13} \pmod{85}$

  - $40^2 = 1\,600 \equiv 70 \pmod{85}$
  - $40^4 = (40^2)^2 \equiv 70^2 \equiv 4\,900 \equiv 55 \pmod{85}$
  - $40^8 = (40^4)^2 \equiv 55^2 \equiv 3\,025 \equiv 50 \pmod{85}$
- $40^{13} \equiv 40^{8+4+1} \equiv 40^8 \times 40^4 \times 40 \equiv 50 \times 55 \times 40 \equiv 10 \pmod{85}$
- On retrouve bien le message $m = 10$ de Bob



Clés d'Alice

- publique : n et e
- privée : d

Lemme

Soit d l'inverse de e modulo $\varphi(n)$ avec $n = p \times q \quad (p \neq q)$

$$\boxed{\text{Si} \quad x \equiv m^e \pmod{n} \quad \text{alors} \quad m \equiv x^d \pmod{n}}$$

Démonstration

- - d est l'inverse de e modulo $\varphi(n)$
  - $d \times e \equiv 1 \pmod{\varphi(n)}$
  - il existe $k \in \mathbb{Z}$ tel que $d \times e = 1 + k \times \varphi(n)$
- Petit théorème de Fermat amélioré

  si $\quad \mathrm{pgcd}(m, n) = 1 \quad$ alors $\quad m^{\varphi(n)} = m^{(p-1)(q-1)} \equiv 1 \pmod{n}$
- Si $\mathrm{pgcd}(m, n) = 1$ alors modulo n :

$$\begin{aligned} (m^e)^d &\equiv m^{1+k\times\varphi(n)} \equiv m \times m^{k\times\varphi(n)} \\ &\equiv m \times (m^{\varphi(n)})^k \equiv m \times (1)^k \\ &\equiv m \pmod{n} \end{aligned}$$

Lemme

Soit d l'inverse de e modulo $\varphi(n)$ avec $n = p \times q \quad (p \neq q)$

$$\boxed{\text{Si} \quad x \equiv m^e \pmod{n} \quad \text{alors} \quad m \equiv x^d \pmod{n}}$$

Démonstration

- Si $\text{pgcd}(m, n) \neq 1$, par exemple $\text{pgcd}(m, n) = p$ et $\text{pgcd}(m, q) = 1$, alors
  - modulo p : $m \equiv 0 \quad \text{et} \quad (m^e)^d \equiv 0 \quad \text{donc} \quad (m^e)^d \equiv m \pmod{p}$
  - modulo q : $(m^e)^d \equiv m \times (m^{\varphi(n)})^k \equiv m \times (m^{q-1})^{(p-1)k} \equiv m \pmod{q}$
  - $\text{pgcd}(p, q) = 1$ $\qquad (m^e)^d \equiv m \pmod{n}$

$\square$

Alice choisit p, q et e, calcule $n = p \times q$ et la clé secrète

Algorithme rsa.py (1)

```
def cle_privee(p,q,e) :
    n = p * q
    phi = (p-1)*(q-1)
    c,d,dd = euclide_etendu(e,phi)   # Pgcd et coeff de Bézout
    return(d % phi)                         # Bon représentant
```

Le chiffrement d'un message m connaissant la clé publique n et e

Algorithme rsa.py (2)

```
def chiffrement_rsa(m,n,e):
    return puissance(m,e,n)
```

Seule Alice peut déchiffrer le message x, à l'aide de sa clé privée d

Algorithme rsa.py (3)

```
def dechiffrement_rsa(x,n,d):
    return puissance(x,d,n)
```

Restent deux problèmes à résoudre :

- Comment construire de grands nombres premiers ?
- Comment calculer "rapidement" l'exponentiation d'un nombre (modulo n) ?