Cryptosystème RSA

Anca Nitulescu
anca.nitulescu@ens.fr

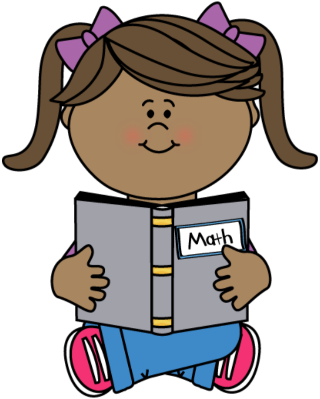
Ecole Normale Supérieure, Paris

Cours 3

Rappels mathématiques

Outils mathématiques

- Algorithme d'Euclide étendu
- Nombres premiers grands
- Exponentiation modulaire
- Inversion modulaire
- Calcul des restes chinois



Arithmétique modulaire

- $(\mathbb{Z}_n, +)$ forme un **groupe additif commutatif d'ordre** n.
- $(\mathbb{Z}_n, +, \cdot)$ forme un **anneau commutatif**.
- **Inverse modulaire** de a dans $\mathbb{Z}_n$: entier $b = a^{-1}$ tel que

$$a \times b = 1 \mod n$$

- $\mathbb{Z}_n^\star =$ l'ensemble des éléments inversibles modulo n.
- $(\mathbb{Z}_n^\star, \cdot)$ forme un **groupe multiplicatif**.
- $(\mathbb{Z}_p, +, \cdot)$ forme un **corps commutatif**.

Attention !

$\mathbb{Z}_n^\star \neq \mathbb{Z}_n \setminus \{0\}$ pour n **composé**
$\mathbb{Z}_p^\star = \mathbb{Z}_p \setminus \{0\}$ pour p **prime**

Critère d'inversibilité

Les entiers inversibles modulo n

$x \in \mathbb{Z}_n^\star$ est inversibles modulo n si et seulement si $\text{pgcd}(x, n) = 1$.

Preuve : T. Bézout.

Calcul de l'inverse modulaire

Trouver $x^{-1} \mod n$

- Il existe u et v tels que $xu + nv = \text{pgcd}(x, n) = 1$
- Trouver l'inverse d'un élément revient à calculer u.
- L'algorithme d'Euclid étendu calcule des coeficients (u, v)

Fonction d'Euler

Définition

- $\varphi(n)$ est le nombre d'entiers de $[1, n]$ qui sont premiers avec n.
- $\varphi(n)$ désigne l'ordre du groupe multiplicatif $\mathbb{Z}_n^\star$

Propriétés

si p est premier et q premier :

- $\varphi(p) = p - 1$
- $\varphi(pq) = \varphi(p)\varphi(q)$

Exponentiation modulaire

Théorème de Lagrange

Si $\mathbb{G}$ est un groupe multiplicatif d'ordre n, alors :

$$\forall g \in \mathbb{G} \quad g^n = e$$

Théorème d'Euler

Pour tout entier n et tout $a \in \mathbb{Z}_n^\star$, on a

$$a^{\varphi(n)} = 1 \mod n$$

Petit théorème de Fermat

Pour p premier et tout entier a on a

$$a^p = a \mod p$$

Exponentiation modulaire

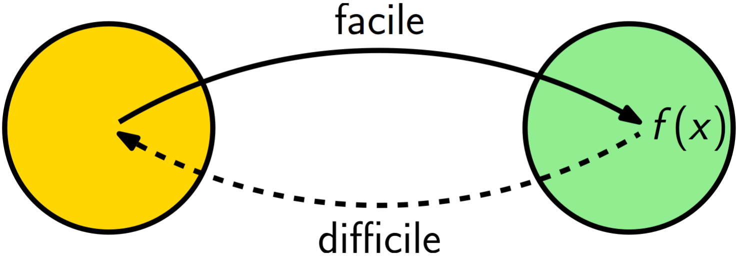
Ordre du groupe $\mathbb{Z}_n^\star$

- ordre $|\mathbb{Z}_n^\star| = \varphi(n) \quad \Rightarrow \quad a^{\varphi(n)} = 1 \pmod n$
- ordre $|\mathbb{Z}_p^\star| = \varphi(p) = p - 1 \Rightarrow a^{p-1} = 1 \pmod n$

Règles

- Dans une exponentiation modulaire (modulo un entier M), les exposants doivent être pris modulo $\varphi(M)$.
- Effectuer les réduction modulaires au fur et à mesure.

Fonctions à sens unique



Fonctions à sens unique

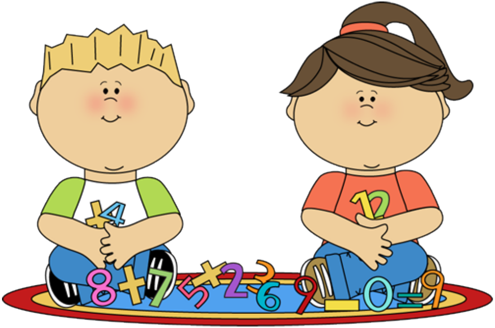
Principe

Pour une relation $y = f(x)$, calculer y est facile, et retrouver x à partir de y est difficile sans une "trappe".

Question

Comment construire telles fonctions ?

Factorisation des entiers



Factorisation

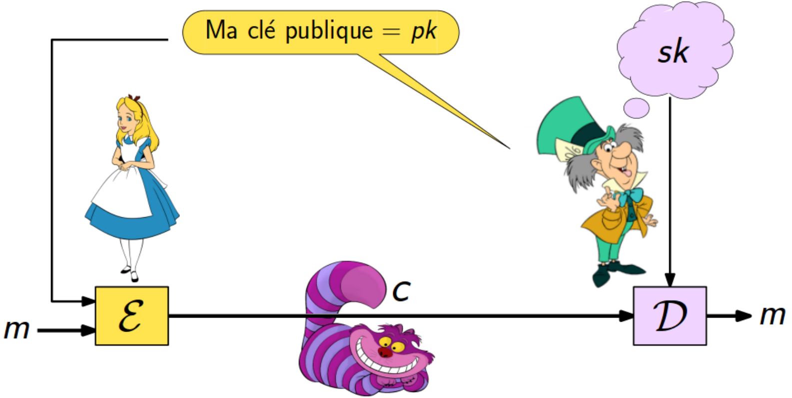
- $(p, q) \rightarrow p \cdot q$ facile
- $n = p \cdot q \rightarrow (p, q)$ difficile

Méthodes connues = exponentielles

Algorithmes de factorisation

- **Divisions successives** $\rightarrow \mathcal{O}(\sqrt{n})$
- **Algorithme de Fermat** : trouver $n = a^2 - b^2 \rightarrow \mathcal{O}(n^{1/3})$
  cas facil : $p - q$ petit
- **Méthode de Gauss** : trouver des résidus quadratiques mod n
  cas facil : $\varphi(n)$ connu
- **Algorithme $p-1$ de Pollard** $\rightarrow \mathcal{O}(p \log(p))$
  cas facil : $p - 1$ et $q - 1$ ont des petits facteurs premiers
- **Algorithme Williams**
  cas facil : $p + 1$ et $q + 1$ ont des petits facteurs premiers
- **Algorithmes sous-exponentiels**
  crible quadratique, courbes elliptiques, crible algébrique

Chiffrement à clé publique



Chiffrement à clé publique

Protocole

- **Algorithme de génération des clés** $\mathcal{KG}(\ell) = (\mathsf{pk}, \mathsf{sk})$
  à partir d'un paramètre de sécurité, il produit une paire de clés

- **Algorithme de chiffrement** $\mathcal{E}(\mathsf{pk}, m) = c$
  produit le chiffré d'un message m, par la clé publique

- **Algorithme de déchiffrement** $\mathcal{D}(\mathsf{sk}, c) = m$
  utilise la clé sécrete/privée sk pour retrouver m à partir de c

Protocole RSA

RSA - Génération des clés

$\mathcal{KG}(\ell) = (\mathsf{pk}, \mathsf{sk})$

- Soit $n = p \cdot q$ (p et q premiers)
- L'ordre du groupe multiplicatif $\mathbb{Z}_n^\star = \varphi(n) = (p-1)(q-1)$
- Soit e un entier premier avec $\varphi(n) = (p-1)(q-1)$
- Soit d un entier qui satisfait $d \cdot e = 1 \pmod{\varphi(n)}$

$$d \cdot e + u\varphi(n) = 1 \qquad \text{(Bézout)}$$

clé publique

- $n = pq$: module public
- e : exposant public

clé secrète

- $d = e^{-1} \pmod{\varphi(n)}$
- les premiers p et q

Protocole RSA

RSA - Chiffrement

$\mathcal{E}(\mathrm{pk} = (e, n), M) = M^e \pmod{\mathrm{n}}$

RSA - Déchiffrement

$\mathcal{D}(\mathrm{sk} = \mathrm{d}, C) = C^d \pmod{\mathrm{n}}$

Vérification

$$(M^e)^d = M^{ed} = M^{1-u\varphi(n)} = M \cdot 1 = M \quad (mod \;\; n)$$

(Théorème d'Euler)

RSA - Exemple simplifié

Deux petits premiers : $p = 5$ et $q = 7$

- $n = 5 \cdot 7 = 35, \quad \varphi(n) = (5-1)\cdot(7-1) = 24$
- e et d : $ed = 1 \mod 24$
  - $ed = 1$: Non, trop petit
  - $ed = 25$: Ok, mais $e = d = 5$ et alors clé privé = clé publique
  - $ed = 49$: Pareil, $e = d$
  - $ed = 73$: 73 est premier, raté
  - $ed = 97$: 97 est premier, raté
  - $ed = 121$: 11 au caré, encore raté
  - $ed = 165$: 165 = 5 * 33, et 5 est premier : Ok
- Clé publique $= (RSA, 35, 5)$ • Clé privée $= (RSA, 33)$.

Conseils d'utilisation du RSA

RSA - Précautions

Il y a de nombreuses manières de **mal utiliser** RSA et d'ouvrir des failles de sécurité !

- Ne jamais utiliser de valeur n trop petite
- Ne jamais utiliser d'exposant e trop petit
- N'utiliser que des clés fortes
  ($p-1$ et $q-1$ ont un grand facteur premier)
- Ne pas chiffrer de blocs trop courts
- Ne pas utiliser de n communs à plusieurs clés

Attaques RSA

RSA - Attaques mathématiques

- factoriser $n = pq$ et par conséquent trouver φ(n) et puis d
- déterminer φ(n) directement et trouver d
- trouver d directement (si petit)
- attaques "broadcast"
- attaques sur modulo n commun
- attaques de synchronisation

(sur le fonctionnement du déchiffrement)

Efficacité de RSA

RSA - coût

Le coût est celui d'une exponentiation modulaire :

Chiffrement : $\mathcal{E}(\mathrm{pk} = (e, n), M) = M^e \pmod{n}$

- $3|e|/2$ multiplications
- si $|n| = |e|$ coût total $1.5 \log^3 n$

Déchiffrement : $\mathcal{D}(\mathrm{sk} = \mathrm{d}, C) = C^d \pmod{n}$

- $3|p|/2$ multiplications mod p $+ 3|q|/2$ multiplications mod q
- $\approx 3|p|$ multiplications mod p
- $\approx 3|n|/8$ multiplications mod p

Propriétés du chiffrement

RSA = Homomorphisme

Le chiffré d'un produit $M_1 M_2$ est égal au produit des chiffrés

$$C_1 = M_1^e \pmod{n}$$

$$C_2 = M_2^e \pmod{n}$$

$$C_1 C_2 = M_1^e M_2^e = (M_1 M_2)^e$$

Intéressant pour certains scénarios
Mais aussi nuisible à la sécurité ...

Attaque à chiffré choisi

1. Soit $C = M^e$ un message chiffré
2. On fabrique $C' = A^e M^e$ le chiffré du message $A \cdot M$
3. Le déchiffrement de C' fournit celui de C

Propriétés du chiffrement

RSA = Déterministe

Chiffrement déterministe = si l'on chiffre plusieurs fois le même message, on obtient le même chiffré

Méthodes pour éviter le chiffrement par substitution :

- couper le message en grands blocs
- modifier la taille de blocs à chaque fois
- randomiser le chiffrement RSA

Sécurité de RSA

RSA - Sécurité

RSA est considéré sécure :

- impossible à déchiffrer sans connaitre l'exposant d de la clé secrète
- trouver la clé secrète d est equivalent à factoriser $n = pq$
- pas d'algorithme polynomial en temps en fonction de la taille des données (la taille des nombres n et e) pour factoriser n
- meilleur algorithme connu est sous-exponentiel (crible algébrique) :

$$\mathcal{O}\left(e^{1.92(\ln n)^{1/3}(\ln \ln n)^{2/3}}\right)$$

Sécurité de RSA

RSA - Sécurité

Pour évaluer et tester la sécurité du RSA :

- évaluer la rapidité des algorithmes de factorisations de grands nombres entiers
- démontrer que la clef secrète de déchiffrement d ne peut pas être obtenue sans factoriser $n = pq$
- montrer que on ne peut pas déchiffrer un message sans la clé secrète d

Problèmes difficiles

Factorisation

- $(p, q) \to p \cdot q$ facile
- $n = p \cdot q \to (p, q)$ difficile

Meilleur algorithme (crible algébrique) : $\mathcal{O}\left(e^{1.92(\ln n)^{1/3}(\ln \ln n)^{2/3}}\right)$

Fonction RSA

Extraction de racine *e*-ième

- $x \to x^e \mod n$ facile
- $y = x^e \to x \mod n$ difficile

avec la trappe $d = e^{-1} \pmod{\varphi(n)}$:

Problèmes difficiles

Factorisation

- $(p, q) \to p \cdot q$ facile
- $n = p \cdot q \to (p, q)$ difficile

Meilleur algorithme (crible algébrique) : $\mathcal{O}\left(e^{1.92(\ln n)^{1/3}(\ln \ln n)^{2/3}}\right)$

Fonction RSA

Extraction de racine e-ième

- $x \to x^{e} \mod n$ facile
- $y = x^{e} \to x \mod n$ difficile

avec la trappe $d = e^{-1} \pmod{\varphi(n)}$: $x = y^{d} = x^{ed} = x \mod n$

Difficulté de RSA

Réduction

Si on connaît la factorisation, on casse RSA :

RSA se réduit à la factorisation !

Le contraire est peut-être faux !

- calculer des racines e-ièmes sans factoriser ? ? ?

En pratique

La factorisation est la seule méthode **connue** pour casser RSA.