Théorie des nombres algorithmique

(Aspects classiques)

2023-2024

Table des matières

1 Formalisme **7**
1.1 Automates finis et langages 7
1.2 Machines de Turing 8
1.3 Exponentiation rapide 8

2 Arithmétique et modules **9**
2.1 Miller-Rabin 9
2.2 Distributions 10
2.3 Densité de premiers 10
2.4 Racines carrées dans $\mathbb{F}_p^\times$ 10
2.5 Carrés dans $(\mathbb{Z}/n\mathbb{Z})^\times$ 10
2.6 Symbole de Jacobi 11
2.7 Un protocole d'identification à divulgation nulle 11

3 Logarithmes discrets **13**
3.1 Réseaux de relations 13
3.2 Algorithmes génériques 13
3.3 Automorphismes de groupe 13
3.4 Un nouveau protocole d'identification à divulgation nulle . . . 13

Introduction

Le cours discute l'algorithmique quantique et le but c'est l'algo de Shor [Sho97] !

Chapitre 1

Formalisme

1.1 Automates finis et langages

Un alphabet est un ensemble de symboles, on regarde en général $\Sigma = \{0, 1\}$ les binaires. Ensuite y'a le langage élémentaire :

$$\{\emptyset, \in, 0, 1\}$$

À partir du langage élémentaire on construit les langages régulier, par concaténations et unions finies.

Définition 1.1.1 (Langage)**.** On prends comme convention que les sous ensembles

$$L \subset \Sigma^*$$

où $\Sigma = \{0, 1\}$.

Pour les automates on prends des 5-tuples $(Q, \Sigma, \delta, q_0, F)$ où Q est un ensemble d'états Σ l'alphabet,

$$\delta \colon Q \times \Sigma \to Q$$

une fonction de transition, q_0 l'état initial et F l'ensemble des états acceptés/terminaux. On étant ensuite δ en

$$\delta^* \colon Q \times \Sigma^* \to Q$$

par $\delta^*(q, w) = \delta^*(\delta(q, w_n), w_0 \ldots w_{n-1}$ avec $w_= w_0 \ldots w_n$. Le truc fun c'est qu'on peut déf le langage accepté par l'automate par :

$$L_\delta = \{w \in \Sigma^* | \delta^*(q_0, w) \in F\}.$$

1.2 Machines de Turing

En gros c'est un automate fini plus une tape infinie à droite et une tête de lecture qui écrit et efface sur la tape.

Définition 1.2.1 (Machine de Turing)**.** Une machine de Turing est un tuple (Σ, K, S, s) avec $S\colon K \times \Sigma \to (\mathbb{K} \cup \{Y, N, H\} \times \Sigma \times \{\bullet, \leftarrow, \rightarrow\}$ où on a les états... Un langage $L \subset \Sigma^*$ est accepté par M ssi $w \in L \leftrightarrow$ la machine s'arrête sur Y.

Définition 1.2.2 (Langage décidable)**.** Un langage est décidable si il existe une machine de Turing qui l'accepte.

Définition 1.2.3 (Fonction récursive)**.** Fonction qui est calculable par une machine de Turing.

Étant donné une fonction $f\colon \mathbb{N} \to \mathbb{N}$.

Définition 1.2.4. On dit qu'une machine de Turing a complexité $O(f)$ si elle termine en temps $f(|n|)$ pour une entrée n de taille $|n|$.

Définition 1.2.5 (PTIME)**.** Dans l'ensemble des langages 2^{Σ^*} on regarde PTIME l'ensemble des langages décidables de complexité polynomial.

Définition 1.2.6 (FPTIME)**.** Dans l'ensemble des fonctions $(\Sigma^*)^{\Sigma^*}$ on déf $FPTIME$ l'analogue pour les fonctions.

1.3 Exponentiation rapide

Algorithm 1 Calcul de $a^e \mod N$

1: Écrire $e = \sum e_i 2^i$.
2: Calculer et enregistrer $a^{2^i} \mod N$ en réduisant à chaque carré par N pour les $e_i \neq 0$.
3: Multiplier $\prod_i a^{2^i} = a^e \mod N$.

Chapitre 2

Arithmétique et modules

Théoreme 2.0.1. *Soit* $r \leq 1$ *et* $M \leq \mathbb{Z}^r$ *alors il existe* $a_1, \ldots, a_s$ *avec* $0 \leq s \leq r$ *et une base* $v_1, \ldots, v_r$ *de* $\mathbb{Z}^r$ *telle que* $a_1 \mid \ldots \mid a_s$ *et*

$$M = \bigoplus_{i}^{s} a_i v_i$$

Grâce à lui on peut résoudre un système linéaire $AX = 0$ en calculant une base de $\ker(A)$.

Théoreme 2.0.2. *Soit* G *un groupe abélien de type fini. Alors il existe* $a_1 \mid \ldots \mid a_s$ *t.q*

$$G \simeq \mathbb{Z}^r \oplus \bigoplus_{i=1}^{r} (\mathbb{Z}/a_i\mathbb{Z})$$

et la décomposition est unique.

2.1 Miller-Rabin

Un nombre $n \geq 2$ est pseudo-premier si $a^{n-1} \equiv 1 \mod n$ pour tout $a \in (\mathbb{Z}/n\mathbb{Z})^\times$. On peut détecter si n est composé par contre.

Théoreme 2.1.1. *Si* n *est impair premier, alors* $n - 1 = 2^k.m$ *et pour tout* $a \in (\mathbb{Z}/n\mathbb{Z})^\times$ *alors* $a^m = 1$ *ou* $a^{m2^c} = -1$ *pour un* $c \leq k$.

Théoreme 2.1.2. *Si* $n \geq 15$ *est composé et impair alors MR(n,a) est faux pour plus de* $\varphi(n)/4$ *éléments de* $(\mathbb{Z}/n\mathbb{Z})^\times$.

2.2 Distributions

Si $X : \Omega \to G$ une variable aléatoire de loi uniforme. Et Y une loi quelconque, alors $Z = X.Y$ suis une loi uniforme!

2.3 Densité de premiers

Soit $A \subset \mathbb{R}$ alors

$$\pi(A) = \#\{n | 1 \leq n \leq A,\ n \text{ est premier}\}$$

a un équivalent

$$\pi(A) = \frac{A}{ln(A)}(1 + o(1))$$

C'est Hadamard, de la vallée Poussin (1900).

2.4 Racines carrées dans $\mathbb{F}_p^\times$

On note $p - 1 = 2 * q$ si $(a, p) = 1$ et $a^q = 1$. On regarde le cas où $p \equiv 3 \mod 4$. Si on note $l \equiv 2^{-1} \mod q$ alors on écrit $q = 2l - 1$. Soit S les carrés de $\mathbb{F}_p^\times$, S est cyclique de taille q impaire. Alors $[2] : S \to S$ est une bijection d'inverse $[l] : S \to S$ (là $[n]$ c'est $[n].g = g^n$). Alors, on a $r \equiv a^l \mod p$ avec $l = \frac{q+1}{2}$ et $r^2 = a \mod p$.

2.5 Carrés dans $(\mathbb{Z}/n\mathbb{Z})^\times$

On note $n = \prod_i p_i^{e_i}$. Par le théorème chinois, $x \mod n$ est un carré ssi $\forall i\ x \mod p_i^{e_i}$ est un carré.

Les cas particuliers $n = pq$ et $n = p^2 q$ sont les cas RSA, et trouver une racine carrées implique de factoriser n avec notre méthode. En fait à l'inverse si on peut calculer des racines carrées, on peut factoriser.

Pour le cas RSA on remarque que $S_n = S_p \times S_q$ et $\# \ker([2]) = 4$ de sorte que $\#S_n = \varphi(n)/4$.

Remarque 1. *Si on a un oracle de racines carrées mod n, on peut prendre des éléments $y = x^2$ pour un x aléatoire et demander une autre racine de y mod n a l'oracle. Comme $\mod n$ y'a plus que 2 racines carrées on obtient un $x' \neq \pm x$ avec bonne probabilité, en fait probabilité $1/2$ si $p \approx q$ via le fait que $x \equiv x' \mod p$ mais pas $\mod q$ ou inversement. Ensuite faut calculer $(x - x') \wedge n$.*

2.6 Symbole de Jacobi

Théoreme 2.6.1. *Soit $m, n \geq 3$ **impairs**. On a*

$$\left(\frac{m}{n}\right) \times \left(\frac{n}{m}\right) = (-1)^{(m-1)(n-1)/4}$$

et $\left(\frac{-1}{m}\right) = (-1)^{(m-1)/2}$, $\left(\frac{2}{m}\right) = (-1)^{(m^2-1)/8}$.

Dans le cas RSA $\left(\frac{x}{n}\right)$ vaut 1 veut seulement dire que x est un carré ou x n'est ni un carré mod q ni mod p. On déf les presques carrés $\{x | \left(\frac{x}{n}\right) = 1\}$. Les carrés sont d'indices 2 dedans. Maintenant on peut faire du bit commitment car chance de 1/2 d'être un carré.

2.7 Un protocole d'identification à divulgation nulle

Protocole d'identification :

- Alice construit son secret $n_A = p_A + q_A$ et choisit $r_A \in (\mathbb{Z}/n_A\mathbb{Z})^{times}$, elle calcule ensuite $r_A^2 = s_A \mod n_A$ un carré aléatoire.
- Elle publie $(Alice, n_A, s_A)$.
- Bob contacte Alice et Alice prouve son identité. Alice choisit un élément aléatoire $u \in (\mathbb{Z}/n\mathbb{Z})^\times$ et calcule son carré $u^2 = v \mod n$ puis $t = v \times s_A$. (s_A, v sont publiques tandis que r_A, u sont secrets.
- Alice envoie v et t (Elle connaît la racine carrée de v et $t = s_A v$.
- Bob génère un bit ϵ pour demander une racine carrée de v ou t. (C'est le twist, on peut pas avoir une racine des deux sinon on peut obtenir r_A.)
- Alice répond et Bob vérifie.
- On peut répeter suffisamment de fois.

Chapitre 3

Logarithmes discrets

3.1 Réseaux de relations

Étant donné G un groupe abélien, soit $g_1, g_2, \ldots, g_I \in G$. Et soit $0 \to R \to \mathbb{Z}^I \to G$ la suite associée, on appelle R le réseau de relations des g_i. Comme G est fini, R est automatiquement de rang plein!

3.2 Algorithmes génériques

Théoreme 3.2.1 (Victor Shoup)**.** *Dans le cas générique, pas d'algorithmes meilleurs que* $\sqrt{\#G}$*.*

Si $G = (\mathbb{Z}/n\mathbb{Z})^\times$ pour n premier, alors c'est plutôt $exp(\sqrt{\log(n)\log\log(n)} \times k)$. Dans le cas des courbes elliptiques sur les corps finis on sait pas.

3.3 Automorphismes de groupe

Si on regarde $G = (\mathbb{Z}/p\mathbb{Z})^d$ alors $Aut(G) = GL_d(\mathbb{F}_p)$ de cardinal $p^{d(d-1)/2} \times \prod_{i=1}^{d}(p^k - 1)$. Pour $(\mathbb{Z}/n\mathbb{Z}, +)$, les automorphismes c'est juste $(\mathbb{Z}/n\mathbb{Z})^\times$ via l'exponentiation. Alors on peut reformuler le log discret en disant : étant donné deux générateurs g, h de G cyclique trouver un automorphisme de G qui envoie g sur h.

3.4 Un nouveau protocole d'identification à divulgation nulle

Protocole :

- Alice génère son secret : Elle choisit un groupe cyclique large, par exemple un s.g de $(\mathbb{Z}/p\mathbb{Z})^{\times})$ ou $E(\mathbb{F}_p)$.
- Elle construit un générateur g_A de G et choisit un a aléatoire de sorte à construire $g_A^a = h_A$.
- Elle publie ensuite $(G, \#G = e, g_A, h_A)$.
- Bob demande à Alice de s'authentifier, Alice choisit alors $a_r \in (\mathbb{Z}/e\mathbb{Z})^{imes}$ et calcule $h_r = h_A^r$ puis envoie h_r.
- Bob demande alors soit un exposant qui envoie h_A vers h_r ou un exposant de g_A vers h_r.

Bibliographie

[Sho97] Peter W. Shor. « Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer ». In : *SIAM Journal on Computing* 26.5 (oct. 1997), p. 1484-1509. issn : 1095-7111. doi : 10.1137/s0097539795293172. url : http://dx.doi.org/10.1137/S0097539795293172.