Master 2 Algèbre Appliquées
Université Paris-Saclay
2016 - 2017

Projet étudiant
Algorithmique et programmation C

COMPTAGE DE POINTS DE COURBES ELLIPTIQUE SUR DES CORPS FINIS

par DANIEL RESENDE

le 18 février 2017

RÉSUMÉ. — Il s'agit de la description de l'algorithme de René Schoof. Celui-ci fût le premier algorithme de comptage de points de courbes elliptique sur des corps finis en un temps polynomial ($O(\log^9 p)$).

Sommaire

Introduction 2
§ 1. Courbes elliptiques sur $\mathbf{F}_q$ 2
§ 2. Algorithme de Schoof 4
§ 2.1. Cas général 4
§ 2.2. Amélioration de Schoof 4
§ 3. Étude de la complexité 6
§ 4. Architecture du programme 6
§ 5. Résultats expérimentaux 6

Introduction

Les courbes elliptiques définissent une lois de groupe sur les corps finis $\mathbf{F}_q$ qui est difficile pour le problème du logarithme discret. On retrouve par conséquent son utilisation dans plusieurs schémas cryptographiques comme Diffie-Hellman (avec ECDH) ou El-Gamal (avec ECDSA). Cependant, l'utilisation de schémas à l'aide de courbes elliptiques nécessite d'avoir un grand nombre premier qui divise l'ordre d'un sous-groupe cyclique de la courbe de $E(\mathbf{F}_q)$. Nous avons donc besoin de connaitre le cardinal de $E(\mathbf{F}_q)$.

FIGURE 1 – Portrait René Schoof

Il existe aujourd'hui de nombres algorithme de comptage de points d'une courbes elliptiques sur un corps finis $\mathbf{F}_q$:

- L'algorithme Baby Step Giant Step basé sur le théorème de Hasse,
- L'algorithme Schoof en 1985 que l'on va étudier dans ce mémoire,
- L'algorithme SEA [Schoof, Elkies, Atkin] en 1995 qui est une amélioration de l'algorithme de Schoof,
- L'algorithme de Satoh en 2005 basé sur le relèvement canonique sur les $\mathbf{Z}$ q-adiques,
- L'algorithme AGM [Mestre] basé sur le calcul de suites arithmetico-géométriques.

Dans ce projet, je vais vous présenter un algorithme de comptage de points de courbes elliptique sur des corps finis. Je me restreindrais à des corps finis $\mathbf{F}_q$ avec $q = p^n$ et p premier différent de 2 et 3. Pour c'est deux derniers cas, l'algorithme est sensiblement le même.

1 Courbes elliptiques sur $\mathbf{F}_q$

Soit $\mathbf{F}_q$ un corps fini à p éléments de caractéristiques $p \neq 2, 3$.
Soit E une courbe elliptique définie sur $\mathbf{F}_q$. On obtient l'équation affine de Weierstraß :

$$y^2 = x^3 + ax + b \tag{1}$$

avec $a, b \in \mathbf{F}_q$ et $\Delta = -16(4a^3 + 27b^2) \neq 0$.

Définition 1.1. Soit Φ l'endomorphisme de Frobénius d'une courbe elliptique E tel que

$$\begin{array}{rccc} \Phi : & E(\bar{\mathbf{F}}_q) & \longrightarrow & E(\bar{\mathbf{F}}_q) \\ & (x, y) & \longmapsto & (x^p, y^p). \end{array}$$

Définition 1.2 (Trace)**.** Soit E une courbe elliptique sur $\mathbf{F}_q$. La trace de $E(\mathbf{F}_q)$ est l'entier $t = q + 1 - \#E(\mathbf{F}_q)$.

PROPOSITION 1.1
Soit la trace t de $E(\mathbf{F}_q)$, on a alors

$$\phi^2 - t\phi + q = 0 \tag{2}$$

THÉORÈME 1.1 (de Hasse)
Soit E une courbe elliptique sur $\mathbf{F}_q$ et la trace t de $E(\mathbf{F}_q)$. On a

$$| t | \leq 2\sqrt{q}, \tag{3}$$

et par conséquent

$$| \#E(\mathbf{F}_q) - (q + 1) | \leq 2\sqrt{q} \tag{4}$$

Nous allons maintenant nous concentrer sur les sous-groupe de n-torsions $E[n]$ avec $n \in \mathbf{Z}_{\geq -1}$ tel que $p \nmid n$. Et on introduit la notion de polynôme de division.

Définition 1.3 (Polynôme de division)**.** Soit $n \in \mathbb{Z}^*$, le polynôme de division ψ_n est la fonction polynôme de $K[E]$ de coefficient dominent n et de diviseur

$$div(\psi_n) = (E[n]) - n^2(\vartheta)$$

PROPOSITION 1.2 (Caractérisation du polynôme de division)
On construit le polynôme de division par récurrence sur $n \in \mathbf{Z}_{\geq 1}$:

1. $\psi_{-1}(X,Y) = -1,\ \psi_0(X,Y) = 0,\ \psi_1(X,Y) = 1,\ \psi_2(X,Y) = 2Y,$
2. $\psi_3(X,Y) = 3X^4 + 6aX^2 + 12bX - a^2,$
3. $\psi_4(X,Y) = 4Y(X^6 + 5aX^4 + 20bX^3 - 5a^2X^2 - 4bX - 8b^2 - a^3,$
4. $\psi_{2n}(X,Y) = \psi_n(\psi_{n+2}\psi_{n-1}^2 - \psi_{n-2}\psi_{n+1}^2)/2Y,$
5. $\psi_{2n+1}(X,Y) = \psi_{n+2}\psi_n^3 - \psi_{n+1}^3\psi_{n-1},$
6. $\psi_{-n} = \psi_n.$

Démonstration. Voir [**?**]. □

Dans l'algorithme de Schoof, nous utiliserons une variante du polynôme de division.

Définition 1.4. Soit $n \in \mathbb{Z}^*$, le polynôme f_n est une fonction polynôme de $K[E]$ définie par les relations suivantes :

$$f(n) = \begin{cases} \bar{\psi_n}(X,Y) & \text{si } n \text{ est pair} \\ \bar{\psi_n}(X,Y)/Y & \text{si } n \text{ est impair} \end{cases}$$

où $\bar{\psi_n}$ est la réduction de ψ_n par les termes en Y^2 par l'équation (E).

PROPOSITION 1.3 (Caractérisation de f_n)
On construit f_n par récurrence sur $n \in \mathbf{Z}_{\geq 1}$:

1. $f_{-1}(X) = -1,\ f_0(X) = 0,\ f_1(X) = 1,\ f_2(X) = 2,$
2. $\psi_3(X) = 3X^4 + 6aX^2 + 12bX - a^2,$
3. $\psi_4(X) = 4Y(X^6 + 5aX^4 + 20bX^3 - 5a^2X^2 - 4bX - 8b^2 - a^3,$
4. $f_{2n}(X,Y) = f_n(f_{n+2}f_{n-1}^2 - f_{n-2}f_{n+1}^2),$
5. $$f(n) = \begin{cases} \bar{\psi_n}(X,Y) & \textit{si n est pair} \\ \bar{\psi_n}(X,Y)/Y & \textit{si n est impair} \end{cases}$$
6. $f_{2n+1}(X,Y) = \psi_{n+2}\psi_n^3 - \psi_{n+1}^3\psi_{n-1},$

Démonstration. Voir [**?**]. □

2 Algorithme de Schoof

2.1 Cas général

Cette algorithme consiste à calculer la trace du frobénius modulo tous les $l < l_{max}$ tel que l_{max} soit le plus grand nombre premier vérifiant :

$$\prod_{l \text{ premier, } p \nmid l}^{l_{max}} l > 4\sqrt{q}. \tag{5}$$

Une fois calculé la trace modulo toutes les l-torsions, on utilise le Théorème des Restes Chinois (CRT) pour obtenir la trace dans $\mathbf{F}_q$. Puis on utilise le théoreme de Hasse pour avoir le cardinal de la courbe E sur $\mathbf{F}_q$.

THÉORÈME 2.1 (Algorithme de Schoof)
Voici le descriptif de l'algorithme de Schoof :

Algorithme 1 Algorithme de Shoof

Require: Une courbe elliptique E sur $\mathbf{F}_q$ un polynôme quelconque.
Ensure: Le cardinal de $E(\mathbf{F}_q)$.

```
M ← 2, l ← 3 ;
S ← {(t mod 2, 2)} ; {Cas pour l = 2}
while M < 4√q do
    k ← q mod l ;
    for τ = 0 to (l−1)/2 do
        if ∀P ∈ E[l], φ²(P) + [k]P = ±[τ]φ(P) then
            S ← S ∪ {(τ, l)} or S ← S ∪ {(−τ, l)} {Selon les cas}
            break ;
        end if
    end for
    M ← M ∗ l ;
    l ← nextprime(l) ; {Donne le prochain nombre premier après l}
end while
∀t ∈ S, trace ← CRT(t) ; {Effectue le théorème des restes chinois}
return q + 1 − trace.
```

Démonstration.

Cas mod 2 Dans ce cas, on cherche les points de 2-torsions, *i.e.*les points spéciaux de la courbes.

$$t = 1 \bmod 2 \Leftrightarrow \#E(\mathbf{F}_q)[2] = 1 \Leftrightarrow X^3+aX+b \text{ est irreductible sur } \mathbf{F}_q \Leftrightarrow pgcd(X^3+aX+b, X^q-X) = 1$$

□

2.2 Amélioration de Schoof

Dans son article original, Schoof (voir [tR85]) propose une amélioration possible de son algorithme.
— Si $\forall P$ nonzéro $\phi_l^2 P = \pm kP$ avec $q \equiv k[l]$
— Sinon on fait le cas général.

Algorithme 2 Algorithme de Shoof amélioré

Require: Une courbe elliptique E sur $\mathbf{F}_q$ un polynôme quelconque.
Ensure: Le cardinal de $E(\mathbf{F}_p)$.

$M \leftarrow 2, l \leftarrow 3$;
$S \leftarrow \{(t \ mod \ 2, 2)\}$; {Cas pour $l = 2$}
while $M < 4\sqrt{q}$ **do**
  $k \leftarrow q \ mod \ l$;
  if $\phi_l^2 P = \pm kP$ **then**
    if $(\frac{k}{l}) = -1$ **then**
      $S \leftarrow S \cup \{(0, l)\}$
    else
      on recherche w tel que $k = w^2 \ mod \ l$
      if $\pm w$ est une valeur propre de ϕ_l **then**
        $S \leftarrow S \cup \{(w, l)\}$ **or** $S \leftarrow S \cup \{(-w, l)\}$ {Selon les cas}
      else
        $S \leftarrow S \cup \{(0, l)\}$
      end if
    end if
  else
    for $\tau = 0$ **to** $\frac{l-1}{2}$ **do**
      if $\forall P \in E[l], \ \phi^2(P) + [k]P = \pm[\tau]\phi(P)$ **then**
        $S \leftarrow S \cup \{(\tau, l)\}$ **or** $S \leftarrow S \cup \{(-\tau, l)\}$ {Selon les cas}
        break ;
      end if
    end for
  end if
  $M \leftarrow M * l$;
  $l \leftarrow \ nextprime(l)$; {Donne le prochain nombre premier après l}
end while
$\forall t \in S, \ trace \leftarrow CRT(t)$; {Effectue le théorème des restes chinois}
return $q + 1 - trace$.

3 Étude de la complexité

4 Architecture du programme

Pour rappel, ce programme est écrit avec le langage `C` et j'utilise la librairie `FLINT`. J'ai choisi de décomposer mon programme en trois parties :

- La fonction `main` qui récupère les arguments au près de l'utilisateur, et qui vérifie que les arguments donne une courbe elliptique sur $\mathbf{F}_q$. Elle se fine en affichant le cardinal de la courbe elliptique.
- La fonction `division_polynomial` remplie un tableau avec tous les polynômes de division. J'ai pris le parti de garder en mémoire tous les polynômes de division dans un tableau dynamique. En effet, on sait à l'avance que l'on aura au plus *lmax* polynômes à calculer et on a besoin à de nombreuses reprise des polynômes de division. De plus, c'est une récursivité à $\frac{n-2}{2}$, $\frac{n-1}{2}$, $\frac{n}{2}$, $\frac{n+1}{2}$ et $\frac{n+2}{2}$ (pour $k = 2n$ ou $k = 2n+1$).

  ```
  void division_polynomial(fq_poly_t *tab, fq_t a, fq_t b,fq_poly_t ecc,
  ulong k, fq_ctx_t fq)
  ENTRÉE :
       Tableau de Fq-polynôme tab et k la taille du tableau
       Entiers a,b tels que E: y² = x³ + ax + b une courbe elliptique sur Fq
       Fq-polynôme eec représentant la courbe elliptique
       Corps fini fq à q éléments
  SORTIE :
       Tableau de Fq-polynôme tab rempli de k polynôme de division
  ```

- La fonction `schoof` crée un tableau de *lmax* polynômes de division (remplie par la fonction `division_polynomial`), puis elle exécute l'algorithme de schoof. Et renvoie le cardinal de la courbe elliptique.

  ```
  void schoof(fmpz_t card, fq_t a, fq_t b, fmpz_t q, fq_ctx_t fq)
  ENTRÉE :
       Entier q premier tel que Fq un corps fini à q éléments
       Entiers a,b tels que E: y² = x³ + ax + b une courbe elliptique sur Fq
  SORTIE :
       Entier card tel que card = #E(Fq)
  ```

Par ailleurs, j'ai implémenté une fonction `fmpz_nextprime` qui renvoi le prochain nombre premier. En effet, cette fonction n'est pas inclut dans la librairie `FLINT`. Il est possible d'optimiser cette fonction, cependant dans notre cas les nombres premiers tester sont de l'ordre de $O(log(q))$.

```
void fmpz_nextprime(fmpz_t rop, fmpz_t op)
ENTRÉE :
     Entier op tel que op > 2
SORTIE :
     Entier rop
```

```
void fmpz_nextprime(fmpz_t rop, fmpz_t op)
{
    fmpz_add_ui(rop, op, 2);
    while(!fmpz_is_prime(rop))
    {
        fmpz_add_ui(rop, rop, 2);
    }
}
```

5 Résultats expérimentaux

Références

[tIF99] SMART N. BLAKE I. F., SEROUSSI G. *Elliptic curves in cryptography*, volume 265. Cambridge university press, 1999.

[tR85] SCHOOF René. Elliptic curves over finite fields and the computation of square roots mod p. *Mathematics of computation*, 44(170) :483–494, 1985.

[tR06] POMERANCE C. CRANDALL R. *Prime numbers : a computational perspective*, volume 182. Springer Science & Business Media, 2006.