Numărul de puncte ale curbelor eliptice

Adrian Manea

16 noiembrie 2019

Cuprins

DE ADĂUGAT/CLARIFICAT

INTRODUCERE

de scris introducere

SECȚIUNEA 1

CURBE ELIPTICE

1.1 Generalități

Formal, curbele eliptice sînt varietăți proiective de dimensiune 1 și gen 1, dar ele pot fi definite și intuitiv, la nivel elementar, folosind forma dată de așa-numita *ecuație Weierstrass*.

Fără a intra în detalii generale privitoare la funcțiile Weierstrass, este suficient să definim o *curbă eliptică* printr-o ecuație de forma:

$$y^2 = x^3 + ax + b,$$

care este *nesingulară*, adică nu conține „colțuri“ (eng. *cusps*) și autointersecții. În funcție de corpul peste care este definită curba eliptică, coeficienții a, b sînt elemente ale corpului respectiv.

Clasificarea curbelor eliptice se face folosind *discriminantul* curbei, care se definește prin:

$$\Delta = -16(4a^3 + 27b^2),$$

care trebuie să fie nenul ca să avem o curbă nesingulară.

Aplicațiile în geometrie algebrică și criptografie sînt facilitate de posibilitatea definirii unei structuri de grup pe mulțimea punctelor de pe o curbă eliptică. Această structură de grup este bine precizată riguros, folosind *divizori* (cf., de exemplu, [Silverman, 2009], III.§2), dar pentru scopurile lucrării prezente va fi suficient să descriem operația de grup intuitiv.

Astfel, fie P și Q două puncte de pe curbă. Putem descrie punctul $P + Q$ astfel: se trasează dreapta care conține cele două puncte și al treilea punct de intersecție al acestei drepte cu curba se definește ca fiind opusul rezultatului.

Formal, avem:

Definiţie 1.1: Fie E o curbă eliptică și fie P, Q două puncte pe E.

Fie L dreapta prin P și Q (dacă $P = Q$, atunci L va fi tangenta în P). Fie R un al treilea punct de intersecție al lui L cu E.

Fie L' dreapta care unește R și $O = [0, 1, 0]$, punctul de la infinit.

Atunci L' intersectează E în R, O și un al treilea punct, care se notează $P + Q$.

Operația este ilustrată în figura 1.1

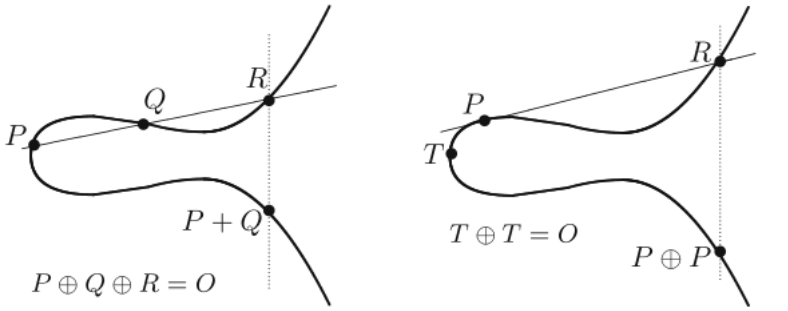

Figura 1.1: *Adunarea punctelor pe o curbă eliptică, [Silverman, 2009], p. 51*

Cu această operație $(E, +)$ formează un grup abelian, cu elementul neutru $O = [0, 1, 0]$.

Exemplu 1.1: Fie curba eliptică E, definită peste $\mathbb{Q}$ prin ecuația Weierstrass:

$$E : y^2 = x^3 + 17.$$

Calcule simple găsesc cîteva puncte cu coordonate întregi:

$$P_1 = (-2, 3), \quad P_2 = (-1, 4), \quad P_3 = (2, 5), \quad P_4 = (4, 9), \quad P_5 = (8, 23).$$

Folosind operația de grup, putem verifica relațiile:

$$P_5 = -2 \cdot P_1, \quad P_4 = P_1 - P_3.$$

Mai general, de fapt, se poate arăta că orice punct rațional $P \in E(\mathbb{Q})$ poate fi scris sub forma:

$$P = mP_1 + nP_3, \quad m, n \in \mathbb{Z},$$

de unde rezultă că $E(\mathbb{Q}) = \mathbb{Z} \times \mathbb{Z}$.

1.2 Aplicația Frobenius

Lucrăm în cazul particular cînd curba E este definită peste un corp finit $\mathbb{F}_q$ (caz dezvoltat și în secțiunile următoare). Așadar, considerăm q o putere a unui prim p, iar $\mathbb{F}_q$ va fi corpul cu q elemente.

Se definește *aplicația Frobenius* prin:

$$\phi : E \longrightarrow E, \quad \phi(x, y) = (x^q, y^q).$$

Mai general, dacă K este un corp care-l extinde pe $\mathbb{F}_q$, se poate defini morfismul Frobenius F în general prin $\alpha \mapsto \alpha^q$.

Cîteva proprietăți elementare, preluate fără demonstrație din [Soeten, 2013]:

Propoziţie 1.1: *Aplicația $F : K \longrightarrow K$ de mai sus satisface proprietățile:*

(a) $F(xy) = F(x)F(y), \forall x, y \in K$;

(b) $F(x + y) = F(x) + F(y), \forall x, y \in K$;

(c) $\mathbb{F}_q = \{\alpha \in K \mid F(\alpha) = \alpha\}$;

(d) Dacă $K = \mathbb{F}_q(t)$ este corpul de funcții raționale peste $\mathbb{F}_q$, într-o nedeterminată t, atunci pentru o funcție rațională $\gamma \in \mathbb{F}_q(t)$ are loc $F(\gamma(t)) = \gamma(t^q)$.

Demonstrațiile sînt manipulări algebrice simple ale proprietăților corpurilor finite, precum și ale caracteristicii q, în mod esențial.

De asemenea, aplicația Frobenius este bijectivă.

poate ceva despre polinoamele de diviziune?

SECȚIUNEA 2

NUMĂRUL DE PUNCTE PESTE $\mathbb{F}_Q$

2.1 Problema și abordarea naivă

Fie acum $E/\mathbb{F}_q$ o curbă eliptică definită peste un corp finit. Vrem să estimăm numărul punctelor din $E(\mathbb{F}_q)$, notat $\#E(\mathbb{F}_q)$, adică una sau mai multe soluții ale ecuației Weierstrass scrisă în forma:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad (x, y) \in \mathbb{F}_q^2.$$

Evident că valoarea lui x conduce la cel mult 2 valori pentru y, deci vom avea o margine superioară:

$$\#E(\mathbb{F}_q) \leq 2q + 1.$$

Dar o ecuație pătratică aleatorie are mici șanse să fie rezolvabilă în $\mathbb{F}_q$, deci ne așteptăm ca marginea superioară să conțină mai curînd q, nu $2q$.

Exemplu 2.1: Să luăm un exemplu simplu mai întîi, pe care să-l rezolvăm manual.

Fie E curba $y^2 = x^3 + x + 1$, definită peste $\mathbb{F}_5$.

Putem calcula direct astfel: luăm resturile lui x modulo 5, calculăm pătratele lor, precum și expresia $x^3 + x + 1$. Urmărind apoi primele două coloane ale tabelului de mai jos, putem găsit dacă există y care să fie pătrat modulo 5, egal cu expresia corespunzătoare.

x	x^2	$x^3 + x + 1$	y	Puncte
0	0	1	1, 4	$(0, 1), (0, 4)$
1	1	3	∄	∄
2	4	1	1, 4	$(2, 1), (2, 4)$
3	4	1	1, 4	$(3, 1), (3, 4)$
4	1	4	2, 3	$(4, 2), (4, 3)$

Rezultă $\#E(\mathbb{F}_5) = 9$, adică cele 8 puncte din tabel și punctul de la infinit.

În acest caz particular, am avut nevoie de $O(5)$ pași, pentru că a trebuit să calculăm toți termenii din tabel, pentru orice $x \in \mathbb{F}_5$.

2.2 Optimizarea 1: Teorema Hasse

Calculul manual de mai sus nu este practic pentru valori mari ale lui q. Avem un rezultat teoretic care ne arată, însă, că nu este necesar să luăm chiar toate valorile $x \in \mathbb{F}_q$, lucru care conduce la faptul că marginea superioară a pașilor de calculat este ceva mai blîndă.

Vom avea nevoie de:

Lemă 2.1: *Fie A un grup abelian și $d : A \longrightarrow \mathbb{Z}$ o formă pătratică pozitiv definită. Atunci:*

$$|d(\psi - \phi) - d(\phi) - d(\psi)| \leq 2\sqrt{d(\phi)d(\psi)}, \quad \forall \psi, \phi \in A.$$

Demonstrație. Fie $\psi, \phi \in A$. Folosim forma biliniară asociată formei pătratice d:

$$L(\psi, \phi) = d(\psi - \phi) - d(\phi) - d(\psi).$$

Cum d este pozitiv definită, pentru orice $m, n \in \mathbb{Z}$:

$$0 \leq d(m\psi - n\phi) = m^2 d(\psi) + mnL(\psi, phi) + n^2 d(\phi).$$

În particular, luăm

$$m = -L(\psi, \phi), \quad n = 2d(\psi)$$

și se obține:

$$0 \leq d(\psi)\left(4d(\psi)d(\phi) - L(\psi, \phi)^2\right).$$

Pentru $\psi \neq 0$, rezultă inegalitatea noastră, iar pentru $\psi = 0$, inegalitatea este trivială. □

Rezultatul important de mai jos a fost formulat ca o conjectură de E. Artin în 1924 și demonstrat de H. Hasse în 1933:

Teoremă 2.1 (Hasse)**:** *Fie $E/\mathbb{F}_q$ o curbă eliptică definită peste corpul finit $\mathbb{F}_q$.*

Atunci:

$$\left|\#E(\mathbb{F}_q) - q - 1\right| \leq 2\sqrt{q}.$$

Demonstrație. Considerăm endomorfismul Frobenius de putere q:

$$\phi : (x, y) \mapsto (x^q, y^q).$$

Din mica teoremă a lui Fermat:

$$x^q \equiv x \bmod q,$$

deci aplicația fixează E punct cu punct, i.e. $\phi(P) = P$, pentru orice punct P de pe curbă.

Rezultă mai departe $(\phi - \mathrm{id})(P) = 0$, adică $P \in \mathrm{Ker}(\phi - \mathrm{id})$. Atunci toată curba E este izomorfă $\mathrm{Ker}(\phi - \mathrm{id})$.

Așadar, $\#E(\mathbb{F}_q)$ coincide cu gradul aplicației $\phi - \mathrm{id}$, ca polinom.

Din lema 2.1, rezultă:

$$|\mathrm{grad}(\phi - \mathrm{id}) - \mathrm{grad}(\phi) - \mathrm{grad}(\mathrm{id}) \leq 2\sqrt{\mathrm{grad}\phi \cdot \mathrm{grad}(\mathrm{id})}.$$

Dar $\mathrm{grad}(\phi - \mathrm{id}) = \#E(\mathbb{F}_q)$, iar $\mathrm{grad}(\phi) = q$ și $\mathrm{grad}(\mathrm{id}) = 1$, rezultă:

$$|\#E(\mathbb{F}_q) - q - 1| \leq 2\sqrt{q}.$$

□

2.3 Optimizarea 2: Baby Step, Giant Step

Putem aplica o metodă simplă de optimizare, schimbînd un pic abordarea.

Fie $P(x, y) \in E(\mathbb{F}_q)$, punct pe care îl alegem luînd valori arbitrare pentru x pînă ce $x^3 + Ax + B$ este un pătrat în $\mathbb{F}_q$. Apoi, calculăm radicalul acestui număr și obținem y. Din teorema lui Hasse (teorema 2.1), știm că:

$$\#E(\mathbb{F}_q) \in (q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q}).$$

Apoi, folosind teorema lui Lagrange din liceu, rezultă că există un punct M în acest interval, cu proprietatea $MP = O$.

Algoritmul nu va funcționa dacă există două astfel de puncte, M și M' astfel încît $MP = M'P = O$. Pentru astfel de cazuri, însă, va fi suficient să repetăm procedura cu un alt punct $P \in E(\mathbb{F}_q)$.

Testarea tuturor valorilor M astfel încît $MP = O$ durează cel mult $4\sqrt{q}$ pași (lungimea intervalului de căutare).

Dar putem îmbunătăți acest număr folosind algoritmul *baby step giant step*, utilizat de obicei pentru calculul logaritmului discret.

Pe scurt, algoritmul este redat în figura 1. Presupunem că se pornește cu un grup ciclic G de ordin n, cu un generator α și un element oarecare β.

Putem acum aplica ideile din algoritmul 1 pentru calculul $\#E(\mathbb{F}_q)$ și să obținem un timp de rulare de aproximativ $4\sqrt[4]{q}$. Algoritmul este redat în figura 2

Cîteva observații importante pe marginea acestui algoritm:

- Am presupus implicit că bucla din liniile 8-10 se execută cel puțin o dată. Într-adevăr, acest lucru rezultă imediat dintr-un rezultat de forma:

  Lemă 2.2: *Fie $a \in \mathbb{Z}$ astfel încît $|a| \leq 2m^2$. Atunci există doi întregi a_0, a_1, cu $-m < a_0 \leq m$ și $-m \leq a_1 \leq m$ astfel încît $a = a_0 + 2ma_1$.*

Algoritm 1 Baby Step, Giant Step

```
1: procedură BSGS(α, β)                                   ▷ returnează log_α β
2:     m ← ⌈√n⌉
3:     pentru 0 ≤ j < m execută
4:         Calculează α^j și notează (j, α^j)
5:     final pentru
6:     Calculează a^{-m}
7:     Definește γ ← β
8:     pentru 0 ≤ i < m execută
9:         dacă γ = α^j pentru un j atunci
10:            returnează i · m + j
11:        altfel
12:            γ ← γ · α^{-m}
13:        final dacă
14:    final pentru
15: final procedură
```

- Calculul $(j+1)P$ atunci cînd se știe jP se poate face mai simplu adunînd P la jP decît prin calculul produsului din nou. Calculul complet necesită, în această variantă, m adunări. Apoi $2mP$ se poate calcula prin dublarea lui mP. Calculul lui Q necesită $\log(q+1)$ dublări și w adunări, unde w este numărul de cifre nenule din reprezentarea binară a lui $q+1$. Mai mult, cunoașterea lui jP și a lui $2mP$ ne permite să reducem numărul de dublări. În fine, pentru a calcula $Q + (k+1)2mP$ din $Q + k(2mP)$, adunăm $2mP$ în loc să calculăm din nou;
- Presupunem implicit că putem factoriza M. Dacă nu, măcar putem găsit toți factorii primi mici p_i și să verificăm $\frac{M}{p_i} \neq O$ pentru aceștia. Atunci M este un candidat bun pentru ordinul lui P, ca element în grupul $E(\mathbb{F}_q)$;
- Linia 20 se bazează pe următorul fapt: cum $MP = O$, ordinul lui P divide M (teorema lui Lagrange de la grupuri). Dacă nu există niciun divizor propriu $\overline{M}$ al lui M cu $\overline{M}P = O$, atunci M este chiar ordinul lui P;
- Algoritmul este costisitor din punct de vedere al spațiului. Se lucrează cu multe numere, unele dintre ele calculate cu proceduri complicate atunci cînd q devine mare. Optimizări ulterioare mai pot folosi, de exemplu, metoda Pollard rho pentru logaritmul discret, în loc de baby step, giant step.

Algoritm 2 Baby Step, Giant Step în $\mathbb{F}_q$

procedură BSGSQ($E(\mathbb{F}_q)$) ▷ returnează $\#E(\mathbb{F}_q)$
    Alegem $m > \sqrt[4]{q}$
    pentru $j = 0, m$ execută
        $P_j \longleftarrow jP$
    final pentru
    $L \longleftarrow q$
    $Q \longleftarrow (q+1)P$
    repetă
        calculează $Q + k \cdot (2mP)$
    până ∃$j, Q + k \cdot (2mP) = \pm P_j$ ▷ comparăm coordonatele x
    $M \longleftarrow q + 1 + 2mk \mp j$ ▷ verificăm $MP = O$
    Factorizăm $M = p_1^{\alpha_1} \cdots p_r^{\alpha_r}$
    cît timp $i \leq r$ execută
        dacă $\frac{M}{p_i} P = O$ atunci
            $M \longleftarrow \frac{M}{p_i}$
        altfel
            $i \longleftarrow i + 1$
        final dacă
    final cît timp
    $L \longleftarrow$ cmmmc(L, M)
    cît timp L divide cel puțin 2 elemente $N \in (q + 1 - 2\sqrt{q}, q + 1 + 2\sqrt{q})$ execută
        Alege un nou punct P și reia de la început
    final cît timp
    returnează $N = \#E(\mathbb{F}_q)$
final **procedură**

SECȚIUNEA 3

ALGORITMUL LUI SCHOOF

de clarificat pașii

de uniformizat descrierea

Există o abordare algoritmică pentru a număra punctele unei curbe eliptice definită peste un corp finit. Știm din teorema lui Hasse (teorema 2.1) că:

$$\#E(\mathbb{F}_q) = q + 1 - a_q, \quad |a_q| \leq 2\sqrt{q}.$$

Pentru aplicații criptografice, însă, este util să avem o metodă eficientă de a calcula numărul de puncte din $E(\mathbb{F}_q)$.

Pentru simplitate, vom presupune că lucrăm cu q impar și că E este dată de o ecuație Weierstrass de forma:

$$E : y^2 = f(x) = 4x^3 + b_2x^2 + 2b_4x + b_6,$$

pentru care mare parte din rezultatele folosite vor fi valabile și în caracteristică 2, cu mici modificări.

Există o metodă directă, dar deloc simplă, de a calcula numărul de puncte, care folosește simboluri Legendre:

$$a_q = \sum_{x \in \mathbb{F}_q} \left(\frac{f(x)}{q} \right),$$

dar fiecare simbol Legendre se calculează folosind reciprocitatea pătratică în $O(\log q)$ pași, deci în total avem $O(q \log q)$ pași, adică un algoritm exponențial.

În continuare, descriem un algoritm care calculează $\#E(\mathbb{F}_q)$ în timp polinomial, i.e. $O(\log^c q)$, cu c fixat, independent de q. Ideea acestui algoritm este să se calculeze $a_q \bmod \ell$ pentru prime mici ℓ și apoi să se folosească lema chineză a resturilor pentru a recompune a_q.

Fie aplicația:

$$\tau : E(\overline{\mathbb{F}}_q) \longrightarrow E(\overline{\mathbb{F}}_q), \quad (x, y) \mapsto (x^q, y^q),$$

aplicația Frobenius de putere q, extinsă la curba definită peste închiderea algebrică a lui $\mathbb{F}_q$, deci știm că are loc:

$$\tau^2 - a_q\tau + q = 0$$

în End(E), conform și demonstrației teoremei 2.1. Se definește subgrupul de ℓ-torsiune al curbei eliptice, pentru un prim impar fixat ℓ:

$$E(\mathbb{F}_q)[\ell] = \left\{ P \in E(\overline{\mathbb{F}}_q) \mid lP = O \right\}.$$

În particular, pentru $P \in E(\mathbb{F}_q)[\ell]$, are loc:

$$\tau^2(P) - [a_q]\tau(P) + [q]P = O,$$

deci dacă punem $P = (x, y)$ și presupunem $P \neq O$, avem:

$$(x^{q^2}, y^{q^2}) - [a_q](x^q, y^q) + [q](x, y) = O.$$

Deoarece am presupus că $P = (x, y)$ are ordinul ℓ, rezultă:

$$[a_q](x^q, y^q) = [n_\ell](x^q, y^q),$$

pentru un $n_\ell \equiv a_q \bmod \ell$ și $0 \leq n_\ell < \ell$.

Similar, putem calcula $[q](x, y)$ prin a reduce q modulo ℓ mai întîi.

Nu trebuie să știm exact valoarea lui n_ℓ, deci pentru orice întreg între 0 și ℓ calculăm $[n](x^q, y^q)$ pentru orice punct $(x, y) \in E[\ell] - \{O\}$ și verificăm dacă satisface:

$$[n](x^q, y^q) = (x^{q^2}, y^{q^2}) + [q](x, y).$$

Problema care apare este că punctele din $E[\ell]$ sînt definite peste extinderi destul de mari ale lui $\mathbb{F}_q$, deci va trebui să lucrăm cu toate punctele de ℓ-torsiune simultan. Pentru aceasta, folosim polinomul $\psi_\ell(x) \in \mathbb{F}_q[x]$, ale cărui rădăcini sînt coordonatele x ale punctelor nenule de ℓ-torsiune din E (presupunem, pentru simplitate, $\ell \neq 2$). Acest polinom are gradul $\frac{1}{2}(\ell^2 - 1)$ și îl descriem pe scurt mai jos.

Definiţie 3.1: Mulțimea polinoamelor de diviziune (eng. *division polynomials*) din $\mathbb{Z}[x, y, A, B]$ se definește recursiv prin:

$$\begin{aligned}
\psi_0 &= 0 \\
\psi_1 &= 1 \\
\psi_2 &= 2y \\
\psi_3 &= 3x^4 + 6Ax^2 + 12Bx - A^2 \\
\psi_4 &= 4y\left(x^6 + 5Ax^4 + 20Bx^3 - 5A^2x^2 - 4ABx - 8B^2 - A^3\right) \\
&\vdots \\
\psi_{2m+1} &= \psi_{m+2}\psi^3 m - \psi_{m-1}\psi_{m+1}^3,\ m \geq 2 \\
\psi_{2m} &= \left(\frac{\psi_m}{2y}\right) \cdot 5\left(\psi_{m+2}\psi_{m-1}^2 - \psi_{m-2}\psi_{m+1}^2\right),\ m \geq 3
\end{aligned}$$

Legătura cu curbele eliptice este realizată astfel. Presupunem că lucrăm cu o curbă eliptică de ecuație $y^2 = x^3 + Ax + B$. Atunci polinoamele de diviziune definite ca mai sus au proprietatea $\psi_{2m+1} \in \mathbb{Z}[x, A, B]$ și $\psi_{2m} \in 2y\mathbb{Z}[x, A, B]$ și în cazul nostru vom avea, suplimentar, $A, B \in \mathbb{F}_q$, deoarece curba este definită peste acest corp.

În plus, rădăcinile polinoamelor ψ_{2n+1} sînt coordonatele x ale punctelor $E(\mathbb{F}_q)[2n+1] - \{O\}$, subgrupul de $2n+1$-torsiune, iar similar, rădăcinile polinoamelor $\dfrac{\psi_{2n}}{y}$ sînt coordonatele x ale punctelor din $E(\mathbb{F}_q)[2n] - E(\mathbb{F}_q)[2]$.

Încă mai mult, $E(\overline{\mathbb{F}}_q) \simeq \mathbb{Z}/\ell \times \mathbb{Z}/\ell$ dacă $\ell \neq p$ și în rest, avem izomorfism cu $\mathbb{Z}/\ell$ sau $\{0\}$ dacă $\ell = p$. Rezultă că, după cazuri, gradul polinomului ψ_ℓ este $\dfrac{1}{2}(\ell^2 - 1)$, $\dfrac{1}{2}(\ell - 1)$ sau 0.

Revenind la algoritmul lui Schoof, acum putem lucra în inelul factor:

$$R_\ell = \frac{\mathbb{F}_q[x, y]}{\psi_\ell(x), y^2 - f(x)}.$$

Rezultă că, dacă avem o putere neliniară a lui y, putem înlocui y^2 cu $f(x)$ și dacă avem o putere x^d, mai mare decît $\dfrac{1}{2}(\ell^2 - 1)$, putem împărți la $\psi_\ell(x)$ și luăm doar restul. Astfel, nu lucrăm niciodată cu polinoame de grad mai mare decît $\dfrac{1}{2}(\ell^2 - 3)$.

Scopul va fi să calculăm a_q mod ℓ pentru suficiente prime ℓ și apoi să găsim a_q. Teorema lui Hasse (2.1) ne dă $|a_q| \leq 2\sqrt{q}$, deci este suficient să luăm primele $\ell \leq \ell_{\max}$ astfel încît:

$$\prod_{\ell \leq \ell_{\max}} \ell \geq 4\sqrt{q}.$$

Teoremă 3.1 (Algoritmul Schoof)**:** *Fie $E/\mathbb{F}_q$ o curbă eliptică. Algoritmul descris la figura 3 este unul în timp polinomial pentru a calcula $\#E(\mathbb{F}_q)$. Mai precis, calculează $\#E(\mathbb{F}_q)$ în $O(\log^8 q)$ pași.*

Demonstrație. Arătăm că timpul de rulare pentru algoritmul Schoof este $O(\log^8 q)$.

Mai întîi:

(a) Cel mai mare număr prim ℓ folosit în algoritm are proprietatea $\ell \leq O(\log q)$:

Teorema de distribuție a numerelor prime poate fi rescrisă în forma:

$$\lim_{x \to \infty} \frac{1}{x} \sum_{\substack{\ell \leq x \\ l \text{ prim}}} \log \ell = 1.$$

Rezultă $\displaystyle\prod_{\ell < x} \ell \simeq e^x$, deci pentru a face ca produsul să fie mai mare decît $4\sqrt{q}$, este suficient să luăm $x \simeq \dfrac{1}{2}\log(16q)$.

Algoritm 3 Algoritmul lui Schoof

1: **procedură** SCHOOF(q, a) ▷ returnează $\#E(\mathbb{F}_q)$
2: $A \longleftarrow 1$
3: $\ell \longleftarrow 3$
4: `cît timp` $A < 4\sqrt{q}$ `execută`
5: `cît timp` $n = 0, 1, 2, \ldots, \ell - 1$ `execută`
6: `dacă` $(x^{q^2}, y^{q^2}) + [q](x, y) = [n](x^q, y^q)$ `atunci ieși`
7: `final dacă`
8: `final cît timp`
9: $A \longleftarrow \ell \cdot A$
10: $n_\ell = n$
11: $\ell \longleftarrow$ următorul prim ℓ
12: `final cît timp`
13: Lema Chineză $\Longrightarrow a \equiv n_\ell \bmod \ell, \forall n_\ell$
14: **returnează** $\#E(\mathbb{F}_q) = q + 1 - a$
15: `final` **procedură**

(b) Înmulțirea în inelul R_ℓ se poate face în $O(\ell^4 \log^2 q)$ operații pe biți:

Elementele inelului R_ℓ sînt polinoame de grad $O(\ell^2)$. Înmulțirea între două astfel de polinoame și reducerea modulo $\psi_\ell(x)$ consumă $O(\ell^4)$ operații elementare (adunări și înmulțiri) în corpul $\mathbb{F}_q$. Similar, înmulțirea în $\mathbb{F}_q$ consumă $O(\log^2 q)$ operați pe biți.

Rezultă că operațiile de bază în R_ℓ consumă $O(\ell^4 \log^2 q)$ operații pe biți.

(c) Sînt necesare $O(\log q)$ operații în inelul R_ℓ pentru a reduce x^q, y^q, x^{q^2} și y^{q^2} în inelul R_ℓ:

În general, sînt necesare $O(\log n)$ operații pentru a calcula puterile x^n și y^n în R_ℓ. Dar aceste operații sînt făcute o singură dată, iar apoi putem stoca punctele de forma:

$$(x^{q^2}, y^{q^2}) + [q \bmod \ell](x, y) \quad \text{și} \quad (x^q, y^q)$$

pe care apoi le folosim în pasul 4 al algoritmului Schoof.

Folosind operațiile elementare de mai sus, putem estima timpul de rulare pentru algoritmul Schoof. Din (a), obținem că avem nevoie doar de ℓ prime care sînt mai mici decît $O(log q)$ și cum există $O\left(\dfrac{\log q}{\log\log q}\right)$ asemenea prime, rezultă că liniile 4-12 din algoritmul lui Schoof se execută de atîtea ori. Apoi, de fiecare dată cînd se intră în bucla controlată de A, se execută bucla controlată de n (liniile 5-8) de $\ell = O(\log q)$ ori.

Mai departe, cum $\ell = O(\log q)$, din afirmația (b) de mai sus, rezultă că operațiile de bază din R_ℓ durează $O(\log^6 q)$ operații pe biți. Valoarea $[n](x^q, y^q)$ din linia 6 a algoritmului se poate calcula în $O(1)$ operații în R_ℓ, știind valoarea anterioară $[n-1](x^q, y^q)$.

Rezultă că numărul total de pași este:

$$\underbrace{O(\log q)}_{\text{bucla A}} \cdot \underbrace{O(\log q)}_{\text{bucla n}} \cdot \underbrace{O(\log^6 q)}_{\text{operații pe biți}} = O(\log^8 q) \text{ operații pe biți.}$$

Am demonstrat, deci, că algoritmul lui Schoof calculează $\#E(\mathbb{F}_q)$ în timp polinomial. □

Remarcăm că cele mai costisitoare etape sînt calculele în inelul R_ℓ, care este o extindere a lui $\mathbb{F}_q$, de grad $2\ell^2$. Așadar, deși marginea pentru ℓ este liniară în $\log q$, pentru valori mari ale lui q, și marginea pentru ℓ și dimensiunea inelului R_ℓ peste $\mathbb{F}_q$ sînt mari.

Exemplu 3.1: Fie $q \simeq 2^{256}$, o valoare utilizată în practică în aplicații criptografice. Rezultă:

$$\prod_{\ell \leq 103} \ell \simeq 2^{133} > 4\sqrt{q} = 2^{130},$$

deci cel mai mare prim ℓ utilizat de algoritmul lui Schoof este $\ell = 103$.

Rezultă că un element din $V = \mathbb{F}_q[x]/\psi_\ell(x)$ este reprezentat de un $\mathbb{F}_q$-vector de mărime $103^2 \simeq 2^{13}$, iar fiecare element al $\mathbb{F}_q$ este un număr pe 256 biți. Așadar, elementele din V ocupă aproximativ 2^{22} biți, adică mai mult de 16 kB. Mărimea nu este nerezonabilă pentru computerele moderne, totuși calcule intensive în inele ale căror elemente se stochează pe 16 kB durează considerabil.

Rezultatul lui Schoof a fost formulat în 1985 și este unul foarte important din punct de vedere teoretic, deoarece toți algoritmii inițiali erau exponențiali, iar cel mai bun de pînă atunci era varianta Baby Step, Giant Step, care are un timp de rulare de asemenea exponențial.

Ulterior, în anii '90, s-au adus îmbunătățiri algoritmului lui Schoof, de către Noam Elkies și A. O. Atkin, prin considerarea unor prime ℓ de o anumită formă și folosind rezultate teoretice mult mai puternice, precum formele modulare.

Varianta îmbunătățită rulează în aproximativ $O(\log^6 q)$, cu presupunerea că aproximativ jumătate dintre primele mai mici decît $O(\log q)$ au proprietatea dorită. Însă această presupunere este una euristică și nu a fost demonstrată, nici măcar în cazul în care ipoteza lui Riemann ar fi adevărată.

BIBLIOGRAFIE

[Husemoller, 2004] Husemoller, D. (2004). *Elliptic Curves*. Springer.

[Silverman, 1994] Silverman, J. (1994). *Advanced Topics in the Arithmetic of Elliptic Curves*. Springer.

[Silverman, 2009] Silverman, J. (2009). *The Arithmetic of Elliptic Curves*. Springer.

[Soeten, 2013] Soeten, M. (2013). Hasse's theorem on elliptic curves. Master's thesis, Rijkuniversiteit Groningen.

[Washington, 2008] Washington, L. (2008). *Elliptic Curves, Number Theory and Cryptography*. Chapman and Hall.