Numărul de puncte ale curbelor eliptice

Adrian Manea

15 noiembrie 2019

Cuprins

DE ADĂUGAT/CLARIFICAT

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

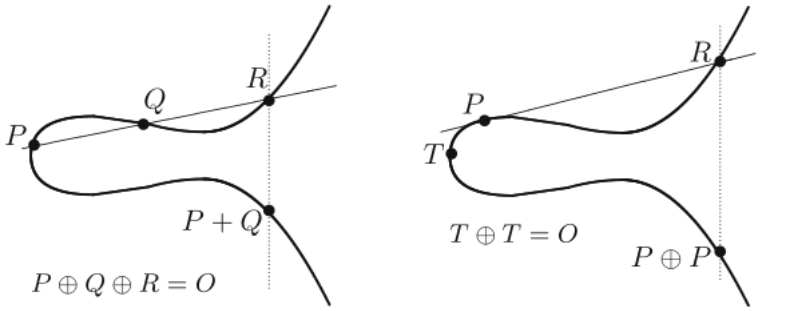

Figura 1.1: *Adunarea punctelor pe o curbă eliptică, [Silverman, 2009], p. 51*

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.

încă generalități...

SECȚIUNEA 2

CURBE ELIPTICE PESTE CORPURI FINITE

Lucrăm acum într-un caz particular, acela al curbelor eliptice definite peste corpuri finite $\mathbb{F}_q$. Notațiile pe care le fixăm sînt:

- q, o putere a unui prim p;
- $\mathbb{F}_q$, un corp finit cu q elemente;
- $\overline{\mathbb{F}}_q$, o închidere algebrică a lui $\mathbb{F}_q$.

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 $2q$.

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:

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

SECȚIUNEA 3

ALGORITMUL LUI SCHOOF

clarifică!

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:

$$\#E(\mathbb{F}_q) = q + 1 - a_1, \quad |a_q| \leq 2\sqrt{q}.$$

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 ecuația 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 mod ℓ 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, deci știm că are loc:

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

în End(E). 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 se poate calcula simplu (**v. Ex. 3.7, pagina 105**). 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 $\frac{1}{2}(\ell^2 - 1)$, putem împărți la $\psi_\ell(x)$ și luăm doar restul. Astfel, nu lucrăm niciodată cu polinoame de grad mai mare decît $\frac{1}{2}(\ell^2 - 3)$.

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 1 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.*

Algorithm 1 Algoritmul lui Schoof

procedure SCHOOF(q, a) ▷ returnează $\#E(\mathbb{F}_q)$
  $A \leftarrow 1$
  $\ell \leftarrow 3$
  while $A < 4\sqrt{q}$ **do**
    while $n = 0, 1, 2, \ldots, \ell - 1$ **do**
      if $(x^{q^2}, y^{q^2}) + [q](x, y) = [n](x^q, y^q)$ **then** `break`
      end if
    end while
    $A \leftarrow \ell \cdot A$
    $n_\ell = n$
    $\ell \leftarrow$ următorul prim ℓ
  end while
  Lema Chineză $\Longrightarrow a \equiv n_\ell \bmod \ell, \forall n_\ell$
  returnează $\#E(\mathbb{F}_q) = q + 1 - a$
end procedure

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 $\ell \preceq O(\log q)$:

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

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

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

(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 $O(logq)$ și cum există $O\left(\frac{\log q}{\log\log q}\right)$ 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: Fie $q \simeq 2^{256}$, o valoare utilizată în practică în aplicații criptografice. Rezultă:

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

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.

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.