

UNIVERZA V LJUBLJANI  
FAKULTETA ZA MATEMATIKO IN FIZIKO

Matematika – 2. stopnja

Miha Avsec

# KUBIČNE KRIVULJE V KRIPTOGRAFIJI

Magistrsko delo

Mentor: doc. dr. Anita Buckley

Somentor: pred. mag. Matjaž Praprotnik

Ljubljana, 2020



# Kazalo

<b>Program dela</b>	<b>v</b>
<b>1 Uvod</b>	<b>1</b>
<b>2 Končna polja</b>	<b>3</b>
<b>3 Kubične krivulje</b>	<b>5</b>
3.1 Točke na krivulji . . . . .	5
3.2 Tonelli-Shanks . . . . .	7
3.3 Struktura grupe na kubičnih krivuljah . . . . .	8
3.4 Endomorfizmi . . . . .	11
3.5 Frobeniusov endomorfizem . . . . .	13
3.6 Red grupe nad eliptičnimi krivuljami . . . . .	17
<b>4 Delitelji</b>	<b>19</b>
<b>5 Torzijske točke</b>	<b>22</b>
<b>6 Weilovo parjenje</b>	<b>26</b>
6.1 Učinkovit algoritem za izračun Weilovega parjenja . . . . .	32
6.2 Implementacija Millerjevega algoritma . . . . .	34
<b>7 Problem diskretnega logaritma</b>	<b>38</b>
7.1 Izračun indeksa . . . . .	40
7.2 Izračun indeksa program . . . . .	42
<b>8 MOV</b>	<b>45</b>
8.1 Mali korak, Velik korak . . . . .	48
8.2 MOV napad program . . . . .	50
<b>9 Anomalne krivulje</b>	<b>52</b>
9.1 Anomalne program . . . . .	56
<b>10 Kriptografija nad Eliptičnimi krivuljami</b>	<b>58</b>
10.1 ElGamalov kriptosistem . . . . .	58
10.2 Kriptosistemi nad parjenji . . . . .	59
<b>11 Zaključek</b>	<b>61</b>
<b>A Program osnovne strukture kubičnih krivulj</b>	<b>61</b>
<b>Literatura</b>	<b>75</b>



## Program dela

## Osnovna literatura

- [18] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*, Chapman and Hall/CRC, druga izd., 2008

Podpis mentorja:



## Kubične krivulje v kriptografiji

### POVZETEK

Skozi delo bomo spoznali osnovne pojme in definicije kubičnih krivulj, ter si ogledali nekaj načinov njihove uporabe v kriptografiji. V primerjavi z drugimi kriptosistemi, kubične krivulje veljajo za bolj varne. Predstavili bomo različne napade na krivulje in na podlagi napadov ocenili, katere krivulje niso oziroma so primerna izbira.

## Elliptic curves in cryptography

### ABSTRACT

Cubic curves in cryptography offer some advantages compared to the other cryptosystems, and they are considered safer if using the same length key. In the master thesis we will first learn some basic definitions and theorems about cubic curves. Then we will present the uses of cubic curves in cryptography, where we will study some cryptosystems and analyze their weaknesses. An important topic discussed will also be safety, which will be studied with multiple attacks on curves. This way we will get a basic idea of which curves are (not) suitable for the use in cryptography.

**Math. Subj. Class. (2010):** 11T71, 94A60, 14H52, 11G20

**Ključne besede:** kubična krivulja, kriptografija, Weilovo parjenje, anomalne krivulje, supersingularne krivulje, MOV napad, index calculus, Millerjev algoritem, torzijske točke

**Keywords:** cubic curve, cryptography, Weil pairing, anomalous curves, supersingular curves, MOV attack, Index calculus, Miller's algorithm, torsion points





# 1 Uvod

Kubične krivulje imajo v kriptografiji velik pomen, saj zagotavljajo enako varnost, kot drugi klasični kriptosistemi, obenem pa potrebujejo manjšo velikost ključa. Njihovo uporabo je prvi predlagal Victor S. Miller leta 1985 [13], a v širšo rabo so vstopile še le okoli leta 2004. Razliko v potrebni dolžini ključa lahko opazujemo na dveh klasičnih zgledih. Kriptosisteme delimo na simetrične in asimetrične. Simetrični kriptosistemi uporabljajo isti ključ za šifriranje in dešifriranje. Pri asimetričnih kriptosistemi pa uporabljamo različna ključa za šifriranje in dešifriranje. Najpogosteje uporabljen asimetrični kriptosistem je RSA. Ime je dobil po svojih avtorjih Rivest, Shamir in Adleman. RSA temelji na problemu faktorizacije števil [8]. Ocenjuje se, da je 2048 bitni ključ v RSA algoritmu enako varen kot 224 bitni ključ nad eliptičnimi krivuljami. Eden najpogosteje uporabljenih simetričnih kriptosistemov pa je AES [8]. Podrobnejšo primerjavo pa lahko razberemo iz spodnje tabele.

<i>AES</i>	<i>ECC</i>	<i>RSA</i>
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	521	15360

Tabela 1: V tabeli so v posamezni vrstici dolžine ključev v bitih, ki zagotavljajo enako varnost, glede na AES, kriptosisteme z uporabo kubičnih krivulj ter RSA [1].

Tu je potrebno dodati, da imajo simetrični kriptosistem pomankljivost pri dogovoru ključa, saj se ne moremo varno dogovoriti za začetni ključ. Klasično se za izmenjavo ključa uporablja npr. RSA ali kriptosistemi nad eliptičnimi krivuljami (na kratko ECC), nato pa se komunikacija nadaljuje s pomočjo simetričnih kriptosistemov. Krajši ključji predstavljajo veliko prednost v okoljih s slabšo procesorsko močjo in/ali omejenim pomnilnikom. Naprave z zgornjimi omejitvami pa so v današnjem svetu kljub hitrim tehnološkim napredkom zelo pogoste. Med njih bi lahko IoT(Internet of Things), pametne kartice, pametne USB ključke, itd. Zgodovinsko gledano sega uporaba kubičnih krivulj za kriptografske namene v konec 20. stoletja. Uporaba kubičnih krivulj pa ni omejena le na kriptosisteme, kubične krivulje namreč lahko služijo tudi kot orodje za napade na klasične kriptosisteme. Tako lahko kubične krivulje uporabljamo pri faktorizaciji števil, s katerim lahko razbijemo vse algoritme, ki temeljijo na faktorizaciji. Pod določenimi pogoji z uporabo kubičnih krivulj dobimo enega najhitrejših poznanih algoritmov za razcep števil [12]. S pomočjo tega lahko napademo vse kriptosisteme, katerih varnost temelji na problemu faktorizacije.

V tem magistrskem delu bomo (enako kot v literaturi) gladkim kubičnim krivuljam pogosto rekli eliptične krivulje, saj so mod  $\mathbb{C}$  vse definicije ekvivalentne. Najprej bomo spoznali osnovne definicije in izreke o kubičnih krivuljah definiranimi nad končnimi polji. Za tem pa se bomo posvetili problemu diskretnega logaritma nad eliptičnimi krivuljami. Nato tega bomo spoznali nekaj napadov na eliptične krivulje, ter analizirali primerno izbiro krivulj za kriptografske namene. Pri tem

bomo uvedli tudi parjenje nad eliptičnimi krivuljami in spoznali učinkovit algoritem za izračun le tega. Za vse skupaj pa bomo napisali tudi programsko kodo.

## 2 Končna polja

Skozi celotno delo se bomo ukvarjali s končimi polji in njihovimi razširitvami. Navedimo najprej vse definicije in lastnosti, ki jih bomo potrebovali.

*Končna polja* ali *Galoisova polja*, so polja s končnim številom elementov. Najosnovnejši primer takih polj so števila modulo  $p$ , kjer je  $p$  praštevilo. Polje s  $q$  elementi označimo z  $\mathbb{F}_q$ , v literaturi pa se pojavlja tudi oznaka  $GF(q)$ . Poglejmo si nekaj pomembnih lastnosti takih polj, ki so povzeta po [4].

### Trditev 2.1.

- *Končno polje s  $q$  elementi obstaja natanko tedaj, ko velja  $q = p^k$ , za nek  $k \in \mathbb{N}$  in neko praštevilo  $p$ .*
- *Vsa končna polja reda  $q$  so si izomorfna.*
- *Vsak element polja  $\mathbb{F}_q$  zadošča enačbi  $x^q - x = 0$ .*
- *Multiplikativna grupa končnega polja je ciklična.*

Poglejmo si, kako konstruiramo polje s  $q = p^n$  elementi. Najprej izberemo nerazcepni polinom  $P$  v  $\mathbb{F}_p[X]$  stopnje  $n$ . Potem velja, da je kvocientni prostor polinomov nad  $\mathbb{F}_p$  z idealom generiranim s  $P$

$$\mathbb{F}_q = \mathbb{F}_p[X]/(P),$$

polje reda  $q$ . Bolj natančno so torej elementi  $\mathbb{F}_q$  polinomi nad poljem  $\mathbb{F}_p$ , stopnje strogo manjše kot  $n$ . Seštevanje in odštevanje poteka na standardni način. Produkt dveh elementov dobimo, kot ostanek pri deljenju s  $P$  produkta polinomov v  $\mathbb{F}_p[X]$ . Inverzne elemente pa lahko poiščemo s pomočjo razširjenega Evklidovega algoritma [3].

### Primer 2.2.

Poglejmo si polje  $\mathbb{F}_4$ . Vzemimo nerazcepni polinom v  $\mathbb{F}_2[x]$

$$X^2 + X + 1.$$

Velja torej

$$\mathbb{F}_4 = \mathbb{F}_2[X]/(X^2 + X + 1) = \{0, 1, \alpha, 1 + \alpha\},$$

kjer je  $\alpha$  ničla zgornjega polinoma v  $\mathbb{F}_4$ . Vse operacije na elementih  $\mathbb{F}_4$  bi lahko zapisali s tabelami

+	0	1	$\alpha$	$1+\alpha$	$\times$	0	1	$\alpha$	$1+\alpha$
0	0	1	$\alpha$	$1+\alpha$	0	0	0	0	0
1	1	0	$1+\alpha$	$\alpha$	1	0	1	$\alpha$	$1+\alpha$
$\alpha$	$\alpha$	$1+\alpha$	0	1	$\alpha$	0	$\alpha$	$1+\alpha$	1
$1+\alpha$	$1+\alpha$	$\alpha$	1	0	$1+\alpha$	0	$\alpha$	1	$1+\alpha$

$x/y$	0	1	$\alpha$	$1+\alpha$
0	/	0	0	0
1	/	1	$1+\alpha$	$\alpha$
$\alpha$	/	$\alpha$	1	$1+\alpha$
$1+\alpha$	/	$1+\alpha$	$\alpha$	1

### Primer 2.3.

Poglejmo si še en zanimiv primer. Naj bo  $\mathbb{F}_q = \mathbb{F}_{p^2}$ , kjer je  $p$  praštevilo za katero velja  $p \equiv 3 \pmod{4}$ . Polinom oblike  $X^2 - r$  je nerazcepen natanko tedaj, ko  $r$  ni kvadratični ostanek po modulu  $p$ . Po Eulerjevem kriteriju [2] je  $r$  kvadratični ostanek natanko tedaj ko  $r^{(p-1)/2} \equiv 1$ . V primeru ko je  $p \equiv 3 \pmod{4}$ ,  $p-1$  ni kvadratični ostanek, saj je  $(p-1)/2 \equiv 1 \pmod{4}$ . To pomeni, da je potenca liha, kar pa nam pove, da  $p-1$  ni kvadratični ostanek. Torej lahko za nerazcepni polinom izberemo  $X^2 + 1$ . Elementi polja  $\mathbb{F}_q$  so torej oblike

$$a + b\alpha,$$

$a, b \in \mathbb{F}_p$ ,  $\alpha$  pa je število za katero velja  $\alpha^2 = -1$ . Ta zapis nas močno spominja na kompleksna števila. Tudi računske operacije se obnašajo enako kot pri kompleksnih številih. V takih poljih lahko torej namesto s polinomi računamo kar s kompleksnimi števili.

V nadaljevanju se bo pogosto pojavljal tudi pojem algebraičnega zaprtja polja  $\mathbb{F}_q$  z oznako  $\overline{\mathbb{F}_q}$ . Spomnimo se potrebnih pojmov za razumevanje le tega.

### Definicija 2.4.

Naj bo polje  $\mathcal{E}$  razširitev polja  $\mathcal{F}$ . Pravimo, da je element  $a \in \mathcal{E}$  *algebraičen* nad  $\mathcal{F}$ , če obstaja neničelni polinom  $f(X) \in \mathcal{F}[X]$ , za katerega velja  $f(a) = 0$ .

### Definicija 2.5.

Polje  $\mathcal{E}$  je *algebraična razširitev* polja  $\mathcal{F}$ , če je vsak element iz  $\mathcal{E}$  algebraičen nad  $\mathcal{F}$ .

**Definicija 2.6.** Polje  $\mathcal{F}$  je *algebraično zaprto*, če ima vsak nekonstanten polinom iz  $\mathcal{F}[X]$  vsaj eno ničlo v  $\mathcal{F}$ .

### Definicija 2.7.

Polje  $\mathcal{A}$  se imenuje *algebraično zaprtje* polja  $\mathcal{F}$ , če je algebraično zaprto in je algebraična razširitev  $\mathcal{F}$ .

### Primer 2.8.

Kompleksna števila  $\mathbb{C}$  so algebraično zaprtje  $\mathbb{R}$ , niso pa algebraično zaprtje  $\mathbb{Q}$ , saj  $\mathbb{C}$  ni algebraična razširitev  $\mathbb{Q}$ .

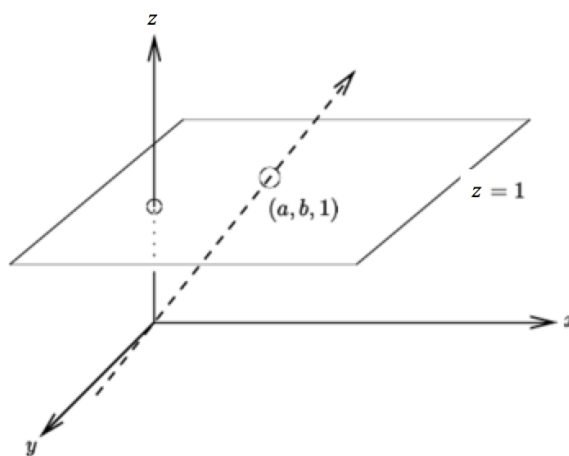
## 3 Kubične krivulje

### 3.1 Točke na krivulji

#### Definicija 3.1.

Projektivna ravnina  $\mathbb{P}^2$  nad poljem  $\mathbb{F}$  je kvocientni prostor  $\mathbb{F}^3 - \{0\}/\sim$ , kjer je ekvivalenčna relacija  $\sim$  podana z  $(a, b, c) \sim (\alpha a, \alpha b, \alpha c)$  za vsak neničelni  $\alpha \in \mathbb{F}$ . Točke v  $\mathbb{P}^2$  so torej podane s homogenimi koordinatami  $[a, b, c] = [\alpha a, \alpha b, \alpha c]$  za vse  $\alpha \neq 0$ .

Točko projektivne ravnine si lahko predstavljamo kot premico skozi izhodišče, kot prikazuje slika 1.



Slika 1: Točka  $[a, b, 1]$  v projektivni ravnini.

#### Definicija 3.2.

Polinom  $P$  je *homogen* stopnje  $d$ , če velja  $P(\lambda x, \lambda y, \lambda z) = \lambda^d P(x, y, z)$  za vse  $\lambda \in \mathbb{F}$ .

#### Definicija 3.3.

*Algebraična krivulja*, podana s homogenim polinomom  $P$ , je množica točk

$$\mathcal{C}_P = \{A \in \mathbb{P}^2, P(A) = 0\}.$$

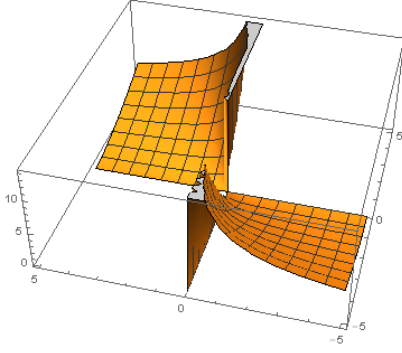
*Kubična krivulja* je algebraična krivulja, podana s homogenim polinomom stopnje 3. V splošnem je polinom oblike

$$a_{300}x^3 + a_{210}x^2y + a_{201}x^2z + a_{120}xy^2 + a_{102}xz^2 + \\ + a_{012}yz^2 + a_{030}y^3 + a_{003}z^3 + a_{111}xyz + a_{021}y^2z,$$

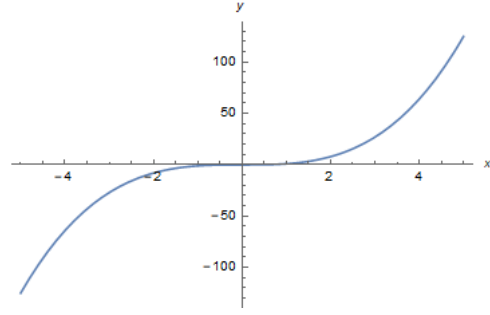
kjer so  $a_{ijk} \in \mathbb{F}$ . Ta zapis vsebuje 10 koeficientov, vendar se lahko v gladkih primerih polinom poenostavi z ustrezno zamenjavo spremenljivk.

#### Definicija 3.4.

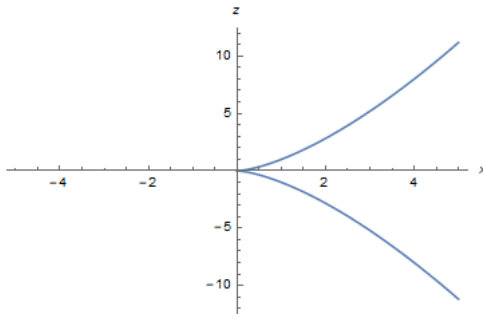
Algebraična krivulja je *gladka*, če nima singularne točke.



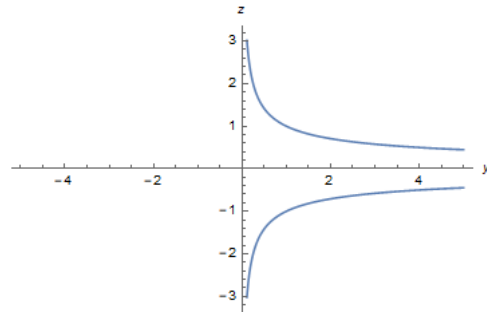
Slika 2: Algebraična krivulja  $\mathcal{C}$  podana s polinomom  $z^2y - x^3$ .



Slika 3: Presek  $\mathcal{C}$  z ravnino  $z = 1$ .



Slika 4: Presek  $\mathcal{C}$  z ravnino  $y = 1$ .



Slika 5: Presek  $\mathcal{C}$  z ravnino  $x = 1$ .

**Izrek 3.5** ([9], Izrek 15.2).

*Enačbo gladke kubične krivulje nad algebraično zaprtim poljem lahko zapišemo v Weierstrassovi obliki*

$$y^2z = x^3 + axz^2 + bz^3.$$

**Opomba 3.6.** Gladki kubični krivulji velikokrat rečemo tudi *eliptična krivulja*.

**Primer 3.7.**

Polinom  $P(x, y, z) = z^2y - x^3$  je homogen polinom stopnje 3. Rešitve enačbe  $z^2y - x^3 = 0$  pa podajajo točke na kubični krivulji.

Na zgornjih slikah lahko vidimo, kako krivuljo predstavimo v projektivni ravnini, ter njene preseke z različnimi afinimi ravninami.

V nadaljevanju nas bodo zanimale predvsem kubične krivulje v končnem polju  $\mathbb{Z}/p\mathbb{Z} \cong \mathbb{F}_p$ , za neko praštevilo  $p$ .

**Definicija 3.8.**

Za dani števili  $a, b \in \mathbb{Z}/p\mathbb{Z}$  je *kubična krivulja* nad poljem  $\mathbb{Z}/p\mathbb{Z}$  množica točk

$$E_{(a,b)}(\mathbb{Z}/p\mathbb{Z}) = \{[x, y, z] \in \mathbb{P}^2(\mathbb{Z}/p\mathbb{Z}) : y^2z = x^3 + axz^2 + bz^3\}.$$

Drugače povedano, afina kubična krivulja je množica rešitev Weierstrassove enačbe

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

Pri čemer upoštevamo zvezo med afinimi in projektivnimi koordinatami točk:

$$(x, y) \in (\mathbb{Z}/p\mathbb{Z})^2 \Leftrightarrow [x, y, 1] \in \mathbb{P}^2(\mathbb{Z}/p\mathbb{Z}).$$

## 3.2 Tonelli-Shanks

V algoritmih bomo velikokrat potrebovali naključno točko na krivulji. S pomočjo algoritma Tonelli-Shanks bomo točko na krivulji lahko učinkovito poiskali. Naj bo  $p$  modul po katerem računamo. Če privzamemo, da lahko naključno izberemo število  $0 \leq x \leq p$ , potem nas zanima, kako učinkovito izberemo naključno točko  $(x, y)$  na krivulji podani z enačbo  $y^2 = x^3 + ax + b \bmod p$ . Ideja je preprosta. Naključno generirajmo koordinato  $x$  in nato poizkusimo rešiti enačbo  $y^2 = X \bmod p$ , če tak  $y$  obstaja. V nasprotnem primeru generiramo novo  $x$  koordinato in poizkusimo ponovno. Problem se torej skriva le v reševanju kvadratne enačbe. Tu pa nam pomaga Tonelli-Shankov algoritem, ki učinkovito reši zgornjo enačbo.

### Trditev 3.9.

*Naj bo  $p > 2$  praštevilo. Če je  $n \in \mathbb{F}_p$  kvadratični ostanek po modulu  $p$ , potem nam Tonelli-Shanks algoritem vrne  $r \in \mathbb{F}_p$ , tako da velja  $n = r^2$ .*

---

#### Algoritem 1 Tonelli-Shanks

---

Vhod: praštevilo  $p$ ,  $n \in \mathbb{F}_p$ .  
 Izhod:  $r \in \mathbb{F}_p$ , za katerega velja  $r^2 = n$ , če ta obstaja  
 Najdi tak  $z \in \mathbb{F}_p$ , da  $z$  ni kvadrat  
 Poišči  $s, Q$ , tako da  $p - 1 = Q2^s$   
 $M = s, c = z^Q, t = n^Q, R = n^{\frac{Q+1}{2}}$   
**while** True **do**  
     **if**  $t = 0$  **then**  
         **return**  $r = 0$   
     **else if**  $t = 1$  **then**  
         **return**  $r = R$   
     **else**  
         Poišči  $0 < i < M$ , da velja  $t^{2^i} = 1$   
          $M = i, c = b^2, t = tb^2, R = Rb$   
     **end if**  
**end while**

---

### Opomba 3.10.

V algoritmu 1 lahko prvi korak izvedemo tako, da preiskujemo naključne  $g \in \mathbb{F}_p$ , dokler ne velja  $g^{(p-1)/2} = -1$ .

**Opomba 3.11.** Števili  $s, Q$  iz algoritma poiščemo tako, da število  $p - 1$  razpolajmo dokler ne dobimo števila, ki ni deljivo z dva. To število je  $Q$ . Število korakov, ki smo jih naredili pa je  $s$ .

Algoritem 1 v povprečju potrebuje

$$2N + 2K + \frac{s(s-1)}{4} + \frac{1}{2^{s-1}} - 9$$

množenj. Tu  $N$  predstavlja število binarnih števk števila  $p$ ,  $K$  pa predstavlja število enic v binarnem zapisu [17].

### 3.3 Struktura grupe na kubičnih krivuljah

Za definicijo grupe na kubičnih krivuljah nad  $\mathbb{C}$  najprej uvedimo pomožno operacijo

$$* : \mathcal{C}_P \times \mathcal{C}_P \rightarrow \mathcal{C}_P,$$

tako da za poljubni točki  $A, B$  na krivulji velja:

$$A * B = \begin{cases} A & \text{če je } A = B \text{ prevoj,} \\ C & \text{če je } \overline{AB} \cap \mathcal{C}_P = \{A, B, C\}, \\ A & \text{če je } \overline{AB} \text{ tangenta v } A, \text{ ter } A \neq B, \\ B & \text{če je } \overline{AB} \text{ tangenta v } B, \text{ ter } A \neq B, \\ C & \text{če je } A = B \text{ in } \{\text{tangenta v } A\} \cap \mathcal{C}_P = \{A, C\}. \end{cases}$$

Intuitivno operacija  $*$  vrne tretjo točko v preseku premice skozi  $A$  in  $B$  in  $\mathcal{C}_P$  kar lahko vidimo na sliki 6. Poglejmo si še nekaj lastnosti operacije  $*$ . Dokaze sledečih trditev najdemo v [9, Poglavje 17.3].

**Trditev 3.12.**

*Operacija  $*$  ima naslednje lastnosti:*

- *komutativnost:*  $A * B = B * A$ ,
- *absorpcija:*  $(A * B) * A = B$ ,
- $((A * B) * C) * D = A * ((B * D) * C)$ .

**Izrek 3.13.**

*Kubična krivulja  $(\mathcal{C}_P, +)$  je Abelova grupa za operacijo*

$$\begin{aligned} + : \mathcal{C}_P \times \mathcal{C}_P &\rightarrow \mathcal{C}_P \\ (A, B) &\mapsto (A * B) * O, \end{aligned}$$

*kjer je  $O$  poljubna izbrana točka na krivulji  $\mathcal{C}_P$ .*

*Dokaz.*

S pomočjo trditve 3.12 dokažimo, da je  $(\mathcal{C}_P, +)$  res Abelova grupa.

- Operacija  $+$  je komutativna:

$$A + B = (A * B) * O = (B * A) * O = B + A.$$

- Točka  $O$  je nevtralni element:

$$A + O = (A * O) * O = A.$$

- Nasprotni element  $A$  definiramo kot  $-A = A * (O * O)$  in preverimo:

$$\begin{aligned} A + (-A) &= (A * (A * (O * O))) * O \\ &= (O * O) * O \\ &= O, \end{aligned}$$

kjer smo uporabili absorpcijo.



- Asociativnost  $(A + B) + C = A + (B + C)$  dokažemo z računom:

$$\begin{aligned}
(A + B) + C &= ((A + B) * C) * O \\
&= (((A * B) * O) * C) * O \\
&= (A * ((B * C) * O)) * O \\
&= (A * (B + C)) * O = A + (B + C). \quad \square
\end{aligned}$$

Ta definicija operacije nudi eleganten geometrijski opis strukture grupe, za numerično računanje pa ni primerna. Možno pa je izpeljati formule, s katerimi eksplicitno izračunamo vsoto dveh točk, v kolikor imamo kubično krivuljo v Weierstrassovi obliki.

**Lema 3.14** (Seštevanje točk na Weierstrassovi kubični krivulji).

Naj bo  $\mathcal{C}_P$  afina krivulja v Weierstrassovi obliki  $y^2 = x^3 + \alpha x^2 + \beta x + \gamma$ , ter  $O$  prevoj v neskončnosti. Če sta  $A_1 = (a_1, b_1)$  in  $A_2 = (a_2, b_2)$  točki na afinem delu  $\mathcal{C}_P$ , potem za  $A_3 = A_1 + A_2 = (a_3, b_3)$  velja

$$\begin{aligned}
a_3 &= \lambda^2 - \alpha - a_1 - a_2 \\
b_3 &= -\lambda a_3 - \mu,
\end{aligned}$$

kjer sta

$$\lambda = \begin{cases} \frac{b_1 - b_2}{a_1 - a_2} & \text{če } a_1 \neq a_2, \\ \frac{3a_1^2 + 2\alpha a_1 + \beta}{2b_1} & \text{sicer,} \end{cases}$$

ter  $\mu = b_1 - \lambda a_1$ .

**Opomba 3.15.**

Če krivuljo  $\mathcal{C}_P$  predstavimo v projektivni ravnini, torej kot ničle homogenega polinomoma  $y^2z = x^3 + \alpha x^2z + \beta xz^2 + \gamma z^3$  je prevoj  $O = [0, 1, 0]$ . Opazimo, da točke  $O = [0, 1, 0]$  ni možno predstaviti v afini ravnini  $z = 1$ , zato bomo v nadaljevanju pisali  $O = \infty$ .

**Primer 3.16.**

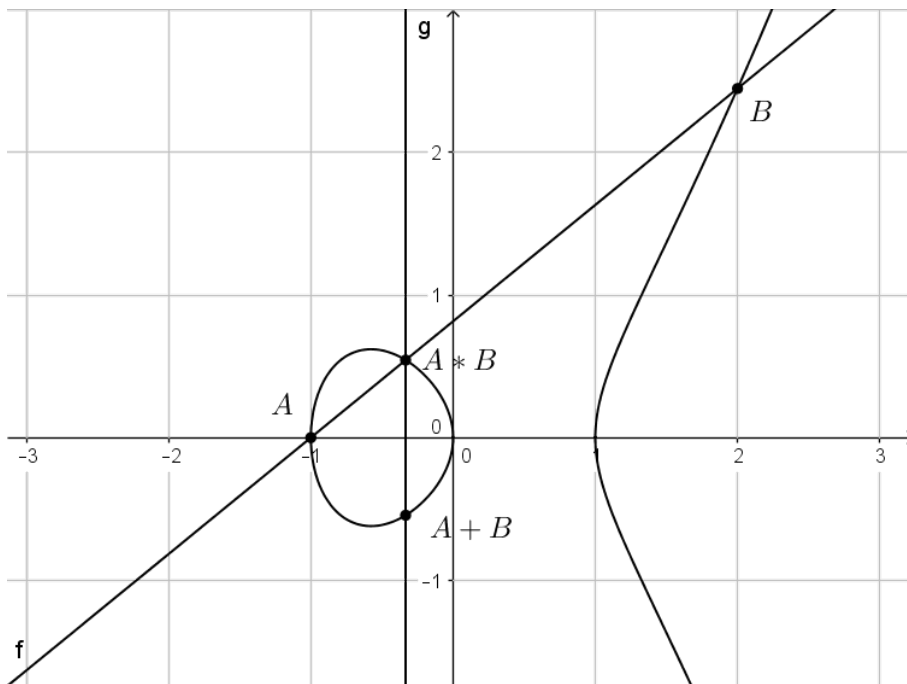
Na sliki 6 je v afini ravnini  $z = 1$  prikazano kako grafično seštevamo točke na Weierstrassovi kubiki  $y^2z - x(x - z)(x + z) = 0$ . Sešteti želimo točki  $A = [-1, 0, 1]$  in  $B = [2, \sqrt{6}, 1]$ .

**Primer 3.17.**

Seštejmo točki  $A = (-1, 0)$ ,  $B = (2, \sqrt{6})$  na Weierstrassovi kubični krivulji  $y^2z - x(x - z)(x + z) = 0$  v preseku z afino ravnino  $z = 1$  še računsko z uporabo zgornje leme 3.14.

Dobimo  $z^2 = x^3 - x$ , torej je  $\alpha = 0$ ,  $\beta = -1$  in  $\gamma = 0$ . Izračunajmo sedaj  $\lambda$  in  $\mu$ , pri čemer upoštevamo prvi predpis, saj sta x-koordinati točk različni:

$$\lambda = \frac{-\sqrt{6}}{-1 - 2} = \frac{\sqrt{6}}{3},$$



Slika 6: Grafično seštevanje točk na kubični krivulji.

$$\mu = 0 - \frac{\sqrt{6}}{3}(-1) = \frac{\sqrt{6}}{3}.$$

Koordinati vsote  $A + B = (x, y)$  sta torej enaki

$$x = \frac{6}{9} - 0 + 1 - 2 = -\frac{1}{3}$$

in

$$y = -\frac{\sqrt{6}}{3}\left(-\frac{1}{3}\right) - \frac{\sqrt{6}}{3} = -\frac{2\sqrt{6}}{9} \doteq -0.5443.$$

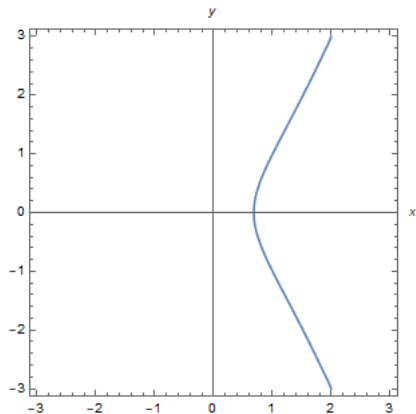
Iskana točka  $A + B \in \mathbb{P}^2$  je torej enaka  $[-\frac{1}{3}, -\frac{2\sqrt{6}}{9}, 1]$ . Dobljeni rezultat se ujema s točko, ki smo jo dobili z grafičnim seštevanjem.

Poglejmo si še primer kubične krivulje nad poljem  $\mathbb{F}_p$ .

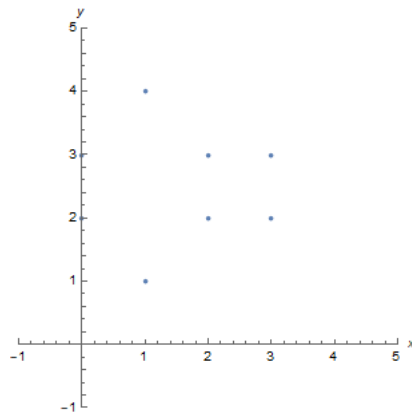
### Primer 3.18.

Naj bo  $E$  krivulja oblike  $y^2 = x^3 + x - 1$  nad poljem  $\mathbb{Z}_5$ . Poglejmo si kako izgledajo točke na krivulji  $E$ .

$x$	$x^3 + x - 1$	$y$	Točke
0	-1	$\pm 3$	$(0, 3), (0, 2)$
1	1	$\pm 1$	$(1, 1), (1, 4)$
2	4	$\pm 3$	$(2, 3), (2, 2)$
3	4	$\pm 3$	$(3, 3), (3, 2)$
4	2	/	/



Slika 7: Algebraična krivulja  $y^2 = x^3 + x - 1$  v  $\mathbb{R}$ .



Slika 8: Algebraična krivulja  $y^2 = x^3 + x - 1$  v  $\mathbb{Z}_5$ .

Izračunajmo sedaj še  $(2, 3) + (1, 1)$ . Uporabimo formule v lemi 3.14, tako dobimo

$$\begin{aligned}\lambda &= \frac{3-1}{2-1} = \frac{2}{1} = 2, \\ x_3 &= 4 - 2 - 1 = 1, \\ \mu &= 3 - 2 \cdot 2 = -1 = 4, \\ y_3 &= -2(1) - 4 = 4.\end{aligned}$$

$(2, 3) + (1, 1)$  je torej enako  $(1, 4)$  na krivulji  $E$ .

### 3.4 Endomorfizmi

#### Definicija 3.19.

Naj bo  $K$  polje nad katerim je definirana eliptična krivulja  $E$ . *Endomorfizem* na  $E$  je homomorfizem  $\alpha : E(\overline{K}) \rightarrow E(\overline{K})$ , ki je podan z racionalno funkcijo. Torej obstajata racionalni funkciji  $R_1$  in  $R_2$  s koeficienti v  $\overline{K}$  za kateri velja

$$\alpha(x, y) = (R_1(x, y), R_2(x, y)),$$

za vse  $(x, y) \in E(\overline{K})$ .

#### Primer 3.20.

Naj bo  $E$  krivulja podana z  $y^2 = x^3 + Ax + B$ , ter naj bo  $\alpha(P) = 2P$ . Tako definiran  $\alpha$  je očitno homomorfizem. Po lemi 3.14 pa obstajajo racionalne funkcije za seštevanje točk na kubičnih krivuljah, zato je  $\alpha$  tudi endomorfizem.

Zaradi različnih možnih oblik racionalnih funkcij bo priročno, če bomo zapis endomorfizmov standardizirali, ter bomo od tod naprej privzeli, da so vsi endomorfizmi zapisani v enotni obliki. V racionalni funkciji  $R(x, y)$  lahko, zato ker se nahajamo na Weierstrassovi krivulji, vse sode potence  $y$  zamenjamo z  $x^3 + Ax + B$  na primerno potenco. Zaradi tega lahko  $R(x, y)$  zapišemo kot

$$R(x, y) = \frac{p_1(x) + p_2(x)y}{p_3(x) + p_4(x)y}.$$

Če sedaj še pomnožimo števec in imenovalc s  $p_3 - p_4y$  ter potem zamenjamo  $y^2$  z  $x^3 + Ax + B$ , dobimo izraz oblike

$$R(x, y) = \frac{q_1(x) + q_2(x)y}{q_3(x)}.$$

Za točke na Weierstrassovi krivulji velja  $-(x, y) = (x, -y)$ , kjer  $-(x, y)$  označuje nasprotni element točke  $(x, y)$ . Od tod sklepamo, da za vsak endomorfizem  $\alpha = (R_1, R_2)$  nad  $E$  velja

$$R_1(x, -y) = R_1(x, y), \quad R_2(x, -y) = -R_2(x, y).$$

To sledi iz dejstva, da je  $\alpha$  homomorfizem in velja

$$\alpha(x, -y) = \alpha(-(x, y)) = -\alpha(x, y).$$

To pa pomeni, da se da  $\alpha(x, y)$  zapisati kot  $\alpha(x, y) = (r_1(x), r_2(x)y)$ , kjer sta  $r_1, r_2$  racionalni funkciji.

Če sedaj zapišemo  $r_1(x)$  kot

$$r_1(x) = \frac{p(x)}{q(x)},$$

potem lahko definiramo še nekaj pojmov.

**Definicija 3.21.**

*Stopnja* endomorfizma je definirana kot

$$\deg(\alpha) = \begin{cases} \max\{\deg p(x), \deg q(x)\} & \text{če } \alpha \neq 0, \\ 0 & \text{če } \alpha \equiv 0. \end{cases}$$

**Definicija 3.22.**

Netrivialni endomorfizem  $\alpha$  je *separabilen*, če je odvod  $r_1'(x) \neq 0$ .

Poglejmo si na primeru, kako določimo stopnjo, ter ali je endomorfizem separabilen.

**Primer 3.23.** Če vzamemo endomorfizem iz prejšnjega primera  $\alpha(P) = 2P$ , dobimo funkcijo

$$R_1(x, y) = \left( \frac{3x^2 + A}{2y} \right)^2 - 2x.$$

To lahko preoblikujemo kot

$$\begin{aligned} \left( \frac{3x^2 + A}{2y} \right)^2 - 2x &= \frac{9x^4 + 6Ax^2 + A^2}{4y^2} - 2x \\ &= \frac{9x^4 + 6Ax^2 + A^2 - 2x(4(x^2 + Ax + B))}{4(x^2 + Ax + B)} \\ &= \frac{x^4 - 2Ax^2 - 8Bx + A^2}{4(x^2 + Ax + B)} \end{aligned}$$

Iz

$$r_1(x) = \frac{x^4 - 2Ax^2 - 8Bx + A^2}{4(x^2 + Ax + B)}$$

razberemo, da je stopnja  $\deg(\alpha) = 4$ .

Poglejmo si še odvod  $r_1'(x)$ .

$$\begin{aligned} r_1'(x) &= \frac{(4x^3 - 4Ax - 8B)4(x^2 + Ax + B) - (x^4 - 2Ax^2 - 8Bx + A^2)(12x^2 + 4A)}{16(x^2 + Ax + B)^2} \\ &= \frac{-A^3 - 8B^2 + 2x^5 - 2A^2x(1+x) + 4Bx^2(2+x) + A(-4Bx + 3x^4)}{4(B + x(A+x))^2} \end{aligned}$$

Števec zgornje enačbe pa ni identično enak 0, torej je endomorfizem separabilen.

### Opomba 3.24.

Če bi zgornje računali v  $\mathbb{F}_2$ , bi dobili primer endomorfizma, ki ni separabilen.

V nadaljevanju bomo potrebovali naslednji izrek, za katerega ne bomo podali dokaza.

### Izrek 3.25 ([18], Izrek 2.21).

Naj bo  $\alpha \neq 0$  separabilni endomorfizem nad eliptično krivuljo  $E$ . Potem velja

$$\deg(\alpha) = \#Ker(\alpha), \text{ kjer } \# \text{ označuje število točk, ki so v jedru } \alpha.$$

Če  $\alpha \neq 0$  ni separabilen, pa velja

$$\deg(\alpha) > \#Ker(\alpha).$$

## 3.5 Frobeniusov endomorfizem

Naj bo  $\mathbb{F}_q$  končno polje z algebraičnim zaprtjem  $\overline{\mathbb{F}_q}$  in naj bo

$$\begin{aligned} \phi_q : \overline{\mathbb{F}_q} &\rightarrow \overline{\mathbb{F}_q}, \\ x &\mapsto x^q \end{aligned}$$

Frobeniusova preslikava na  $\mathbb{F}_q$ . Če je eliptična krivulja  $E$  definirana nad  $\mathbb{F}_q$ , potem  $\phi_q$  deluje na točkah  $E$  kot:

$$\phi_q(x, y) = (x^q, y^q) \text{ in } \phi_q(\infty) = \infty.$$

### Lema 3.26.

Naj bo  $E$  eliptična krivulja definirana nad  $\mathbb{F}_q$ . Za točke  $(x, y) \in E(\overline{\mathbb{F}_q})$  velja

- $\phi_q(x, y) \in E(\overline{\mathbb{F}_q})$ ,
- $(x, y) \in E(\mathbb{F}_q)$  natanko tedaj ko je  $\phi_q(x, y) = (x, y)$ .

*Dokaz.*

Za dokaz leme bomo potrebovali lastnost

$$(a + b)^q = a^q + b^q,$$

kjer je  $q$  potenca karakteristike polja v katerem delamo. To sledi iz razvoja v vrsto in dejstva, da pri binomskih koeficientih velja  $\binom{q}{i} \equiv 0$ , za  $1 \leq i \leq q-1$ . To je res ker pri zapisu binomskega koeficienta po krajšanju vedno ostane vsaj en  $q$ , ki je potenca karakteristike.

Prav tako bomo potrebovali trditev 2.1, da za vse  $a \in \mathbb{F}_q$  velja  $a^q = a$ . Dokaz tega sledi iz dejstva, da ima grupa vseh obrnljivih elementov  $\mathbb{F}_q^\times$  red  $q-1$ , kar pomeni  $a^{q-1} = 1$ , za vse  $a \in \mathbb{F}_q \neq 0$ . To pa je ekvivalentno zgornji trditvi.

Lemo lahko brez težav dokažemo za krivulje v posplošeni Weierstrassovi obliki.

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6, \quad (3.1)$$

kjer so  $a_i \in \mathbb{F}_q$ . Če to enačbo potenciramo na stopnjo  $q$  in uporabimo zgornja dejstva, dobimo

$$(y^q)^2 + a_1(x^qy^q) + a_3(y^q) = (x^q)^3 + a_2(x^q)^2 + a_4(x^q) + a_6.$$

To pa ravno pomeni, da točke oblike  $(x^q, y^q)$  ležijo na krivulji podani z enačbo 3.1.

Za drugi del leme uporabimo dejstvo, da velja  $x \in \mathbb{F}_q$  natanko tedaj, ko je  $\phi_q(x) = x$ . Smer iz leve v desno smo že dokazali zgoraj. Potrebujemo še dokaz v drugo smer. Ta pa sledi iz lastnosti, da ima polinom  $X^q - X$  natanko  $q$  različnih ničel v  $\overline{\mathbb{F}_p}$ . Ker pa množici

$$\{\alpha \in \overline{\mathbb{F}_p} \mid \alpha^q = \alpha\} \text{ in } \mathbb{F}_q$$

vsebujeta  $q$  elementov in je  $\mathbb{F}_q$  vsebovana v drugi množici, sledi da sta enaki.

Od tod za točke  $(x, y) \in E(\mathbb{F}_q)$  sledi

$$\begin{aligned} (x, y) \in E(\mathbb{F}_q) &\Leftrightarrow x, y \in \mathbb{F}_q \\ &\Leftrightarrow \phi_q(x) = x \text{ in } \phi_q(y) = y \\ &\Leftrightarrow \phi_q(x, y) = (x, y). \end{aligned}$$

□

### **Trditev 3.27.**

*Naj bo  $E$  eliptična krivulja definirana nad  $\mathbb{F}_q$ . Naj bosta  $\alpha, \beta$  endomorfizma na  $E$  ter  $a, b \in \mathbb{Z}$ . Definirajmo endomorfizem*

$$(a\alpha + b\beta)(P) = a\alpha(P) + b\beta(P).$$

*Potem velja*

$$\deg(a\alpha + b\beta) = a^2 \deg(\alpha) + b^2 \deg(\beta) + ab(\deg(\alpha + \beta) - \deg(\alpha) - \deg(\beta)).$$

**Opomba 3.28.**

Naj  $n \in \mathbb{N}$  ne deli karakteristike  $K$ . Ker velja  $E[n] \cong \mathbb{Z}_n \oplus \mathbb{Z}_n$ , lahko za  $E[n]$  izberemo neko bazo  $\{\beta_1, \beta_2\}$ . To pomeni, da lahko vsak element  $E[n]$  zapišemo, kot  $m_1\beta_1 + m_2\beta_2$ , za  $m_1, m_2 \in \mathbb{Z}_n$ . Homomorfizem  $\alpha : E(\overline{K}) \rightarrow E(\overline{K})$  preslika  $E[n]$  v  $E[n]$ . Zato obstajajo števila  $a, b, c, d \in \mathbb{Z}_n$ , da velja

$$\alpha(\beta_1) = a\beta_1 + b\beta_2, \quad \alpha(\beta_2) = c\beta_1 + d\beta_2.$$

To pa ravno pomeni, da se lahko vsak tak homomorfizem  $\alpha$  predstavimo z matriko oblike

$$\alpha_n = \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

*Dokaz.*

Naj bo  $n \in \mathbb{Z}$ , število ki ni deljivo s karakteristiko polja. Predstavimo  $\alpha, \beta$  z matrikami  $\alpha_n, \beta_n$ , glede na neko bazo  $E[n]$ . Velja

$$\det(a\alpha_n + b\beta_n) = a^2 \det(\alpha_n) + b^2 \det(\beta_n) + ab(\det(\alpha_n + \beta_n) - \det(\alpha_n) - \det(\beta_n))$$

za vse matrike  $\alpha_n, \beta_n$ . Od tod sledi

$$\begin{aligned} \deg(a\alpha + b\beta) &\equiv a^2 \deg(\alpha) + b^2 \deg(\beta) + ab(\deg(\alpha + \beta) \\ &\quad - \deg(\alpha) - \deg(\beta)) \pmod{n}. \end{aligned}$$

Ker to drži za neskončno mnogo  $n$  v tej enačbi velja enakost.

□

**Trditev 3.29** ([18], Trditev 4.7).

*Naj bo  $E$  definirana nad  $\mathbb{F}_q$  in naj bo  $n \geq 1$ . Potem velja*

- $\text{Ker}(\phi_q^n - 1) = E(\mathbb{F}_{q^n})$ .
- Endomorfizem  $\phi_q^n - 1$  je separabilen. Velja torej  $\#E(\mathbb{F}_{q^n}) = \deg(\phi_q^n - 1)$ .

**Opomba 3.30.**

$\phi_q^n$  predstavlja kompozicijo  $\phi_q \circ \phi_q \circ \dots \circ \phi_q$ . Prav tako pa je  $\phi_q^n - 1$  endomorfizem, saj je množenje z  $-1$  endomorfizem.

**Izrek 3.31** (Hasse [18], Izrek 4.2).

*Naj bo  $E$  eliptična krivulja nad končnim poljem  $\mathbb{F}_q$ . Potem red  $E(\mathbb{F}_q)$  zadošča zvezi*

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

**Opomba 3.32.** Hassejev izrek igra pomembno vlogo tudi pri iskanju velikosti grupe, saj nam poda zgornjo in spodnjo mejo. Velikost grupe pa potem nadaljno zožimo z dejstvom da red elementa deli red grupe.

**Lema 3.33.**

*Naj bosta  $r, s \in \mathbb{Z}$  tuji števili. Potem velja*

$$\deg(r\phi_q - s) = r^2q + s^2 - rsa.$$

*Dokaz.*

Z uporabo trditve 3.27 dobimo

$$\deg(r\phi_q - s) = r^2 \deg(\phi_q) + s^2 \deg(-1) + rs(\deg(\phi_q - 1) - \deg(\phi_q) - \deg(-1)).$$

Če sedaj uporabimo dejstvo  $\deg(\phi_q) = q$ ,  $\deg(-1) = 1$  ter  $\deg(\phi_q - 1) = q + 1 - a$ , željeni rezultat sledi.  $\square$

*Dokaz izreka 3.31.*

Po trditvi 3.29 lahko zapišemo

$$a = q + 1 - \#E(\mathbb{F}_q) = q + 1 - \deg(\phi_q - 1).$$

Pokazati moramo  $|a| \leq 2\sqrt{q}$ . Ker velja  $\deg(r\phi_q - s) \geq 0$ , nam lema 3.33 implicira

$$q \left(\frac{r}{s}\right)^2 - a\frac{r}{s} + 1 \geq 0,$$

za vse  $r, s$  za katere je  $\gcd(s, q) = 1$ . Množica racionalnih števil oblike  $r/s$ , ki zadoščajo tej lastnosti je gosta v  $\mathbb{R}$ . To lahko vidimo tako, da za  $s$  vzamemo potenco števila 2 ali 3. Vsaj eno od teh števil bo tuje  $q$ , saj je  $q$  moč končnega polja in je torej oblike  $q = t^n$ , za neko praštevilo  $t$ . Števila oblike  $\frac{r}{2^m}$  ali  $\frac{r}{3^m}$  pa so očitno gosta v  $\mathbb{R}$ . Zato velja

$$qx^2 - ax + 1 \geq 0,$$

za vse  $x \in \mathbb{R}$ . To pa pomeni, da je diskriminanta polinoma negativna ali nič. Kar povedano z drugimi besedami pomeni

$$a^2 - 4q \leq 0.$$

To pa je ravno pogoj  $|a| \leq 2\sqrt{q}$ .  $\square$

**Izrek 3.34** ([18], Izrek 4.10).

*Naj bo  $E$  eliptična krivulja definirana nad  $\mathbb{F}_q$ . Naj bo  $a = q + 1 - \#E(\mathbb{F}_q) = q + 1 - \deg(\phi_q - 1)$ . Potem velja*

$$\phi_q^2 - a\phi_q + q = 0,$$

*gledano kot endomorfizem nad  $E$ . Prav tako pa je  $a$  edino število  $k$  za katerega velja*

$$\phi_{q^n}^2 - k\phi_{q^n} + q^n = 0.$$

*Povedano drugače, velja naslednja lastnost. Naj bo  $(x, y) \in \overline{\mathbb{F}_q}$ , potem velja*

$$(x^{q^2}, y^{q^2}) - a(x^q, y^q) + q(x, y) = \infty,$$

*obenem pa je  $a$  edino število, tako da ta lastnost velja za vse  $(x, y) \in \overline{\mathbb{F}_q}$ . Velja še več*

$$a \equiv \text{Sled}((\phi_q)_m) \pmod{m},$$

*za vse  $m$ , ki zadoščajo  $\gcd(m, q) = 1$ .*



### 3.6 Red grupe nad eliptičnimi krivuljami

Pogosto nas zanima, koliko točk leži na dani krivulji  $E$ . S pomočjo naslednje trditve bomo dobili preprost način kako izračunati  $\#E(\mathbb{F}_{q^n})$ , če poznamo  $\#E(\mathbb{F}_q)$ . Preostane še vprašanje kako izračunati  $\#E(\mathbb{F}_q)$ . Enega od možnih načinov bomo opisali v opombi 8.7, kot posledico algoritma Velik korak-majhen korak.

**Trditev 3.35.**

Naj bo  $\#E(\mathbb{F}_q) = q + 1 - a$ , kot v izreku 3.34. Zapišimo  $X^2 - aX + q = (X - \alpha)(x - \beta)$ . Potem velja

$$\#E(\mathbb{F}_{q^n}) = q^n + 1 - (\alpha^n + \beta^n),$$

za vse  $n \geq 1$ .

Preden lahko dokažemo zgornjo trditev bomo potrebovali še naslednjo lemo.

**Lema 3.36.**

Naj bo  $s_n = \alpha^n + \beta^n$ . Potem velja  $s_0 = 2, s_1 = a$  in  $s_{n+1} = as_n - qs_{n-1}$ , za vse  $n \geq 1$ .

*Dokaz.*

Če je  $n = 0$ , potem lema očitno velja. Za  $n = 1$  lema sledi iz definicije  $\alpha$  in  $\beta$ , ki sta ničli enačbe  $X^2 - aX + q$ , ter uporabe Vietovih formul. Dokažimo lemo še za splošen  $n$ . Ker sta  $\alpha$  in  $\beta$  ničli iste enačbe, lahko zapišemo  $\alpha^2 - a\alpha + q = 0$  in  $\beta^2 - a\beta + q = 0$ . Sedaj enačbi pomnožimo z  $\alpha^{n-1}$  in  $\beta^{n-1}$ . S tem dobimo

$$\alpha^{n+1} = a\alpha^n + q\alpha^{n-1} \text{ in } \beta^{n+1} = a\beta^n + q\beta^{n-1}.$$

Če enačbi seštejemo, dobimo rekurzivno zvezo

$$s_{n+1} = as_n - qs_{n-1}.$$

□

*Dokaz trditve 3.35.*

Iz leme 3.36 takoj sledi, da velja  $\alpha^n + \beta^n \in \mathbb{N}$ . Naj bo  $f$  funkcija

$$f(X) = (X^n - \alpha^n)(X^n - \beta^n) = X^{2n} - (\alpha^n + \beta^n)X^n + q^n.$$

Iz prve enakosti sledi, da funkcija  $X^2 - aX + q = (X - \alpha)(X - \beta)$  deli  $f$ . Po osnovnem izreku od deljenju lahko  $f$  zapišemo kot  $f(X) = g(X)Q(x) + r(x)$ , kjer je  $g(X) = X^2 - aX + q$ . Prav tako pa ima kvocient  $Q$  celoštevilске koeficiente. Po izreku 3.34 velja

$$(\phi_q^n)^2 - (\alpha^n + \beta^n)\phi_q^n + q^n = f(\phi_q) = Q(\phi_q)(\phi_q^2 - a\phi_q + q) = 0,$$

kjer je  $\phi_q$  endomorfizem na  $E$ . Velja tudi  $\phi_q^n = \phi_{q^n}$ . Ponovno uporabimo izrek 3.34 od koder sledi, da obstaja natanko en  $k \in \mathbb{Z}$ , za katerega je  $\phi_{q^n}^2 - k\phi_{q^n} + q^n = 0$ , in sicer,  $k = q^n + 1 - \#E(\mathbb{F}_{q^n})$ . Od tod torej sledi

$$\alpha^n + \beta^n = q^n + 1 - \#E(\mathbb{F}_{q^n}).$$

□

**Primer 3.37.**

Naj bo  $E$  posplošena Weierstrassova krivulja podana s predpisom  $y^2 + xy = x^3 + 1$ . S preprostim pregledom vseh možnosti vidimo, da je  $\#E(\mathbb{F}_2) = 4$ . Izračunajmo sedaj  $\#E(\mathbb{F}_4)$ . Po trditvi 3.35 moramo  $\#E(\mathbb{F}_2)$  zapisati kot  $q + 1 - a$ . Sledi torej  $a = 2 + 1 - 4 = -1$ . Polinom je potem oblike

$$X^2 + X + 2 = \left(X - \frac{-1 + \sqrt{-7}}{2}\right) \left(X - \frac{-1 - \sqrt{-7}}{2}\right).$$

Za število točk na krivulji moramo poračunati še  $s_2 = \alpha^2 + \beta^2$ , pri čemer si lahko pomagamo z lemo 3.36.

$$s_2 = as_1 - qs_0 = (-1)^2 - 2 \cdot 2 = -3$$

Od tod sledi

$$\#E(\mathbb{F}_{2^2}) = 2^2 + 1 - s_2 = 4 + 1 + 3 = 8.$$

Če naštejemo vse točke na  $E(\mathbb{F}_4) = \{\infty, (0, 1), (1, 0), (1, 1), (1, 2), (3, 0), (0, 3), (1, 3)\}$ , vidimo da je naš rezultat pravilen. Moč trditve 3.35 pa se skriva v velikih potencah. Če bi na primer želeli izračunati  $E(\mathbb{F}_{2^{200}})$  z naštevanjem točk ne bi prišli prav daleč. S pomočjo zgornje trditve in leme pa lahko hitro poračunamo

$$\left(\frac{-1 + \sqrt{-7}}{2}\right)^{200} + \left(\frac{-1 - \sqrt{-7}}{2}\right)^{200} = -2534943693362688758337590430751,$$

od koder sledi

$$\begin{aligned} E(\mathbb{F}_{2^{200}}) &= 2^{200} + 1 + 2534943693362688758337590430751 \\ &= 1606938044258990275541962092343697546215565682541130425732128. \end{aligned}$$

## 4 Delitelji

Pri konstrukciji Weilovega parjenja bodo pomembno vlogo igrali tako imenovani delitelji. V tem poglavju bomo navedli nekaj definicij in lastnosti, ki jih bomo potrebovali v kasnejših poglavjih. Uporabljali bomo terminologijo, ki jo lahko najdemo v [18].

### Definicija 4.1.

Naj bo  $K$  polje in naj bo  $P$  točka na krivulji  $E(\overline{K})$ . Za vsako točko  $P$  definirajmo formalen simbol  $[P]$ . *Delitelj*  $D$  na krivulji  $E$  je končna linearna kombinacija takih simbolov s celoštevilskimi koeficienti.

$$D = \sum_j a_j [P_j], \quad a_j \in \mathbb{Z}$$

Iz same definicije sledi, da je delitelj element Abelove grupe generirane s simboli  $[P]$ . Označimo to grupo z  $\text{Div}(E)$ .

### Definicija 4.2.

Definirajmo *vsoto* in *stopnjo* delitelja kot

$$\text{sum}\left(\sum_j a_j [P_j]\right) = \sum_j a_j P_j \in E(\overline{K}),$$

$$\deg\left(\sum_j a_j [P_j]\right) = \sum_j a_j \in \mathbb{Z}.$$

### Definicija 4.3.

Naj bo  $E$  eliptična krivulja nad poljem  $K$ . *Funkcija* na  $E$  je racionalna funkcija

$$f(x, y) \in \overline{K},$$

ki je definirana za vsaj eno točko na  $E(\overline{K})$ . Funkcija torej zavzame vrednosti v  $\overline{K}$ .

### Opomba 4.4.

Naj bo  $E$  podana z enačbo  $y^2 = x^3 + Ax + B$ . Racionalna funkcija  $\frac{1}{y^2 - x^3 - Ax - B}$  torej ne predstavlja funkcije na  $E$ , saj ni definirana za nobeno točko na  $E$ .

### Trditev 4.5 ([15], Trditev 4.3).

Naj bo  $P$  točka na krivulji  $E$ . Potem obstaja funkcija  $u_P$ , kateri rečemo *uniformalizator*, z lastnostjo  $u_P(P) = 0$ , za katero velja, da lahko vsako funkcijo  $f(x, y)$  nad  $E$  zapišemo kot

$$f = u_P^r g, \quad \text{za nek } r \in \mathbb{Z} \text{ kjer } g(P) \neq 0 \text{ in } \frac{1}{g(P)} \neq 0.$$

### Definicija 4.6.

Številu  $r$  iz trditve 4.5 rečemo *red* funkcije  $f$  v točki  $P$  in ga označimo z  $\text{ord}_P(f)$

**Primer 4.7.**

Naj bo  $y^2 = x^3 - x$  eliptična krivulja, naj bo  $f(x, y) = x$ . Izberimo  $u(x, y) = y$ . Očitno je  $u(0, 0) = 0$ . Nad eliptično krivuljo velja

$$y^2 = x^3 - x = x(x^2 - 1),$$

od tod sledi  $x = y^2 \frac{1}{x^2 - 1}$  nad  $E$ . Prav tako velja  $1/(x^2 - 1) \neq 0$  v točki  $(0, 0)$ . Od tod sledi, da je

$$\text{ord}_{(0,0)}(x) = 2, \quad \text{ord}_{(0,0)}(x/y) = 1.$$

**Definicija 4.8.**

Naj bo  $f$  funkcija nad  $E$ , ki ni identično enaka 0. Definirajmo *delitelja* funkcije  $f$  kot,

$$\text{div}(f) = \sum_{P \in E(\bar{K})} \text{ord}_P(f)[P] \in \text{Div}(E).$$

**Trditev 4.9** ([18], Trditev 11.1).

Naj bo  $E$  eliptična krivulja in naj bo  $f$  funkcija na  $E$ , ki ni identično enaka 0. Potem veljajo naslednje trditve:

- $f$  ima le končno mnogo ničel in polov
- $\deg(\text{Div}(f)) = 0$
- Če  $f$  nima ničel ali polov, potem je  $f$  konstantna.

**Izrek 4.10.**

Naj bo  $E$  eliptična krivulja. Naj bo  $D$  delitelj nad  $E$  z  $\deg(D) = 0$ . Potem obstaja taka funkcija  $f$  na  $E$  z lastnostjo

$$\text{div}(f) = D$$

natanko tedaj ko

$$\text{sum}(D) = \infty.$$

*Dokaz.*

Pokažimo najprej, da se da  $[P_1] + [P_2]$  zapisati kot  $[P_1 + P_2] + [\infty]$  plus delitelj neke funkcije. Recimo, da imamo tri točke  $P_1, P_2, P_3$  na neki krivulji  $E$ , ki ležijo na premici  $ax + by + c = 0$ . Naj bo  $f(x, y) = ax + by + c$ . Potem ima funkcija  $f$  ničle v točkah  $P_1, P_2, P_3$ . Če  $b$  ni enak 0 potem ima funkcija po trditvi 4.9 trojni pol v  $\infty$ . Velja torej

$$\text{div}(ax + by + c) = [P_1] + [P_2] + [P_3] - 3[\infty].$$

Ker se nahajamo na krivulji zapisani v Weierstrassovi obliki, kjer točko  $-P$  dobimo tako, da samo zamenjamo predznak y-koordinate, lahko za premico skozi točki  $P_3 = (x_3, y_3)$ , ter  $-P_3 = (x_3, -y_3)$  vzamemo  $x - x_3 = 0$ . Po trditvi 4.9 ponovno velja

$$\text{div}(x - x_3) = [P_3] + [-P_3] - 2[\infty].$$

Od tod sledi

$$\text{div}\left(\frac{ax + by + c}{x - x_3}\right) = \text{div}(ax + by + c) - \text{div}(x - x_3) = [P_1] + [P_2] - [-P_3] - [\infty].$$

Ker na krivulji velja  $P_1 + P_2 = -P_3$  (to sledi iz načina kako na krivulji seštevamo točke), lahko zgornjo enačbo prepišemo v

$$[P_1] + [P_2] = [P_1 + P_2] + [\infty] + \operatorname{div}(g).$$

Hitro se vidi, da velja

$$\operatorname{sum}(\operatorname{div}(g)) = P_1 + P_2 - (P_1 + P_2) - \infty = \infty.$$

Prav tako, pa se iz zgornje enačbe vidi, da velja  $[P_1] + [P_2] = 2[\infty] + \operatorname{div}(h)$ , če velja  $P_1 + P_2 = \infty$ . Zaradi tega je vsota vseh členov s pozitivnimi koeficienti v  $D$  enaka nekemu simbolu  $[P]$ , večkratniku  $[\infty]$ , ter delitelju neke funkcije. Podobno velja tudi za člene z negativnimi koeficienti. Od tod sledi

$$D = [P] - [Q] + n[\infty] + \operatorname{div}(g_1).$$

Zaradi tega, ker je  $g_1$  kvocient produkta funkcij, ki sestavljajo  $g$ , velja tudi  $\operatorname{sum}(\operatorname{div}(g_1)) = \infty$ . Po trditvi 4.9 velja  $\deg(\operatorname{div}(g_1)) = 0$ , ker funkcija ni konstantna. Imamo torej

$$0 = \deg(D) = 1 - 1 + n + 0 = n.$$

Od tod sledi

$$D = [P] - [Q] + \operatorname{div}(g_1).$$

Prav tako velja

$$\operatorname{sum}(D) = P - Q + \operatorname{sum}(\operatorname{div}(g_1)) = P - Q.$$

Predpostavimo sedaj, da velja  $\operatorname{sum}(D) = \infty$ . Potem  $P - Q = \infty$ , kar pomeni da mora veljati  $P = Q$  in  $D = \operatorname{div}(g_1)$ . Če predpostavimo  $D = \operatorname{div}(f)$  za neko funkcijo  $f$ , potem

$$[P] - [Q] = \operatorname{div}(f/g_1).$$

Od tod po lemi 4.11 sledi  $P = Q$  in torej  $\operatorname{sum}(D) = \infty$ .

□

**Lema 4.11** ([18], Lema 11.3).

*Naj bosta  $P, Q \in E(\overline{K})$ , ter naj obstaja funkcija  $h$  na  $E$  za katero velja*

$$\operatorname{div}(h) = [P] - [Q].$$

*Potem sledi  $P = Q$ .*

## 5 Torzijske točke

### Definicija 5.1.

Naj bo  $E$  eliptična krivulja nad poljem  $K$ , ter naj bo  $n \in \mathbb{N}$ . *Torzijske točke* reda  $n$ , so točke v množici

$$E[n] = \{P \in E(\overline{K}) | nP = \infty\}.$$

### Izrek 5.2.

Naj bo  $E$  eliptična krivulja nad poljem  $K$  in naj bo  $n \in \mathbb{N}$ . Če karakteristika polja  $K$  ne deli  $n$ , ali je enaka 0 potem

$$E[n] \cong \mathbb{Z}_n \oplus \mathbb{Z}_n.$$

Zapišimo  $n = p^r n'$ , kjer  $p \nmid n'$ . Če je karakteristika  $K$  enaka  $p > 0$  in  $p | n$  potem velja

$$E[n] \cong \mathbb{Z}'_n \oplus \mathbb{Z}'_n \text{ ali } E[n] \cong \mathbb{Z}_n \oplus \mathbb{Z}'_n.$$

### Definicija 5.3.

Definirajmo *deliteljski polinom*  $\gamma_m \in \mathbb{Z}[x, y, A, B]$  kot,

$$\begin{aligned} \gamma_0 &= 0 \\ \gamma_1 &= 1 \\ \gamma_2 &= 2y \\ \gamma_3 &= 3x^4 + 6Ax^2 + 12Bx - A^2 \\ \gamma_4 &= 4y(x^6 + 5Ax^4 + 20Bx^3 - 5A^2x^2 - 4ABx - 8B^2 - A^3) \\ \gamma_{2m+1} &= \gamma_{m+2}\gamma_m^3 - \gamma_{m-1}\gamma_{m+1}^3 \text{ za } m \geq 2 \\ \gamma_{2m} &= (2y)^{-1}\gamma_m(\gamma_{m+2}\gamma_{m-1}^2 - \gamma_{m-2}\gamma_{m+1}^2) \text{ za } m \geq 3 \end{aligned}$$

### Lema 5.4.

$\gamma_n$  je element  $\mathbb{Z}[x, y^2, A, B]$ , za vse lihe  $n$ . Za sode  $n$  pa je  $\gamma_n$  element  $2y\mathbb{Z}[x, y^2, A, B]$ .

*Dokaz.*

Dokažimo to s pomočjo indukcije. Za  $n \leq 4$  lema očitno velja. Obravnavajmo primera, ko je  $n = 2m$  in  $n = 2m + 1$  za nek  $m \in \mathbb{N}$ .

- $n=2m$  Indukcijska predpostavka je v tem primeru, da lema velja za vse  $n < 2m$ . Predpostavimo lahko, da je  $2m > 4$ , saj vemo da lema velja za  $n \leq 4$ , torej velja  $m > 2$ . Potem velja  $2m > m + 2$ , kar pomeni, da vsi polinomi v definiciji  $\gamma_{2m}$  zadoščajo indukcijski predpostavki. Če je  $m$  sodo število, potem se  $\gamma_m, \gamma_{m+2}, \gamma_{m-2}$  nahajajo v  $2y\mathbb{Z}[x, y^2, A, B]$ . Od tod pa sledi, da je tudi  $\gamma_{2m} \in 2y\mathbb{Z}[x, y^2, A, B]$ . Če je  $m$  lih, potem sta  $\gamma_{m-1}, \gamma_{m+1} \in 2y\mathbb{Z}[x, y^2, A, B]$ . To pa pomeni, da je tudi  $\gamma_{2m} \in 2y\mathbb{Z}[x, y^2, A, B]$ .
- $n=2m+1$  Primer obravnavamo podobno kot  $n = 2m$ .

□

Definirajmo še polinoma

$$\begin{aligned}\phi_m &= x\gamma_m^2 - \gamma_{m+1}\gamma_{m-1}, \\ \omega_m &= (4y)^{-1}(\gamma_{m+2}\gamma_{m-1}^2 - \gamma_{m-2}\gamma_{m+1}^2).\end{aligned}$$

Podobno, kot pri polinomih  $\gamma$  lahko tudi za  $\phi_m$  in  $\omega_m$  formuliramo lemo, katere dokaz bomo izpustili.

**Lema 5.5.**

$\phi_n$  je element  $\mathbb{Z}[x, y^2, A, B]$ , za vse  $n$ . Če je  $n$  lih, potem je  $\omega_n$  element  $y\mathbb{Z}[x, y^2, A, B]$ . V primeru, da je  $n$  sod pa je  $\omega_n$  element  $\mathbb{Z}[x, y^2, A, B]$ .

Glejmo sedaj te polinome nad eliptičnimi krivuljami oblike

$$E : y^2 = x^3 + Ax + B.$$

Polinome v  $\mathbb{Z}[x, y^2, A, B]$  lahko gledamo kot polinome v  $\mathbb{Z}[x, A, B]$ , tako da  $y^2$  nadomestimo z  $x^3 + Ax + B$ .

**Opomba 5.6.**

Polinoma  $\gamma_n$  ni nujno možno predstaviti, kot polinom spremenljivke  $x$ , saj je potenca  $y$  odvisna od tega ali je  $n$  lih ali sod. Vseeno pa velja, da lahko  $\gamma_n^2$  vedno zapišemo kot polinom spremenljivke  $x$ .

**Izrek 5.7** ([18], Izrek 3.6).

Naj bo  $P = (x, y)$  točka na krivulji  $y^2 = x^3 + Ax + B$ , katere karakteristika ni 2. Naj bo  $n \in \mathbb{N}$ , potem velja

$${}_nP = \left( \frac{\phi_n(x)}{\gamma_n^2(x)}, \frac{\omega_n(x, y)}{\gamma_n^3(x, y)} \right).$$

**Posledica 5.8.**

Naj bo  $E$  eliptična krivulja. Endomorfizem  $E$  podan kot množenjem z  $n$  ima stopnjo  $n^2$ .

Brez dokazov formulirajmo še nekaj pomožnih trditev, ki jih bomo potrebovali za dokaz izreka 5.2.

**Trditev 5.9** ([18], Trditev 2.21).

Naj bo  $\alpha \neq 0$  seperabilen avtomorfizem eliptične krivulje  $E$ . Potem velja

$$\deg \alpha = \#Ker(\alpha),$$

kjer je  $Ker(\alpha)$  jedro homomorfizma  $\alpha : E(\overline{K}) \rightarrow E(\overline{K})$ . Če  $\alpha \neq 0$  ni seperabilen, potem

$$\deg \alpha > \#Ker(\alpha).$$

**Izrek 5.10** ([18], Izrek 2.22).

Naj bo  $E$  eliptična krivulja definirana nad poljem  $K$ . Naj bo  $\alpha \neq 0$  endomorfizem  $E$ . Potem je  $\alpha : E(\overline{K}) \rightarrow E(\overline{K})$  surjektivna preslikava.

**Trditev 5.11** ([18], Trditev 2.28).

Naj bo  $E$  eliptična krivulja definirana nad poljem  $K$ , ter naj bo  $n \in \mathbb{N}$ . Naj bo množenje z  $n$  na  $E$  podano, kot

$$n(x, y) = (R_n(x), yS_n(x))$$

za vse  $(x, y) \in E(\overline{K})$ . Tu sta  $R_n, S_n$  racionalni funkciji. Potem velja

$$\frac{R'_n(x)}{S_n(x)} = n.$$

Od tod pa sledi, da je množenje z  $n$  separabilno natanko tedaj, ko  $n$  ni večkratnik karakteristike polja  $K$ .

Sedaj imamo vse kar potrebujemo, da dokažemo izrek 5.2.

*Dokaz izreka 5.2.*

Predpostavimo, da  $n$  ni večkratnik karakteristike  $p$  polja. Po izreku 5.7 vemo, da ima množenje z  $n$  prvo koordinato oblike

$$R(x) = \frac{x^{n^2} + \dots}{n^2 x^{n^2-1} + \dots}.$$

Če poračunamo odvod dobimo v števcu  $n^2 x^{2n^2-2} + \dots$ , kar ni identično enako 0. To pomeni, da je množenje z  $n$  separabilno. Po posledici 5.8 in trditvi 5.9 ima jedro množenja z  $n$ ,  $E[n]$ , red  $n^2$ . Iz algebre vemo [[7], Posledica izreka 75], da so končne Abelove grupe, torej tudi  $E[n]$  izomorfne

$$\mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2} \oplus \dots \oplus \mathbb{Z}_{n_k},$$

za neka naravna števila  $n_1, n_2, \dots, n_k$ , za katere velja  $n_i | n_{i+1}$  za vse  $i$ . Naj bo  $l$  praštevilo, ki deli  $n_1$ . Potem  $l | n_i$  za vse  $i$ . To pa pomeni, da ima  $E[l] \subseteq E[n]$  red  $l^k$ . Ker ima  $E[l]$  red  $l^2$  po tem kar smo povedali zgoraj, od tod sledi  $k = 2$ . Množenje z  $n$  ima torej jedro  $E[n] \simeq \mathbb{Z}_{n_1} \oplus \mathbb{Z}_{n_2}$ . Veljati pa mora tudi  $n_2 | n$ . Ker velja  $n^2 = \#E[n] = n_1 n_2$  od tod sledi  $n_1 = n_2 = n$ . Zato velja

$$E[n] \simeq \mathbb{Z}_n \oplus \mathbb{Z}_n,$$

ko karakteristika polja ne deli  $n$ .

Pokazati moramo trditev še v primeru, ko  $p | n$ . Po trditvi 5.11 množenje z  $p$  ni separabilno. Po trditvi 5.9 ima jedro  $E[p]$  množenja s  $p$  red strogo manjši kot je stopnja endomorfizma, ki je  $p^2$  po posledici 5.8. Ker ima vsak element  $E[p]$  red 1 ali  $p$ , sledi da ima  $E[p]$  red potence  $p$ . Torej mora biti red 1 ali  $p$ . Če je  $E[p]$  trivialna, potem je  $E[p^k]$  trivialna za vse  $k$ . Če ima  $E[p]$  red  $p$  potem trdimo, da velja  $E[p^k] \simeq \mathbb{Z}_{p^k}$  za vse  $k$ . Pokazati moramo, da ima taka grupa red  $p^k$  in ne nekaj manjšega. Denimo da obstaja element  $P$  reda  $p^j$ . Po izreku 5.10 je množenje z  $p$  surjektivno. Torej obstaja točka  $Q$  tako da velja  $pQ = P$ . Ker velja

$$p^j Q = p^{j-1} P \neq \infty, \quad p^{j+1} Q = p^j P = \infty,$$



torej ima  $Q$  red  $p^{j+1}$ . Z indukcijo lahko pokažemo, da obstajajo točke reda  $p^k$  za vse  $k$ . Ker velja  $E[p^i] \subsetneq E[p^j]$ , za vse  $i < j$ , v  $E[p^k]$  obstaja točka reda  $p^k$ . To pa ravno pomeni, da je  $E[p^k]$  ciklična. Zato je  $E[p^k]$  ciklična reda  $p^k$ .

Zapišimo sedaj  $n = p^r n'$ ,  $r \geq 0$  in  $p \nmid n'$ . Potem velja

$$E[n] \simeq E[n'] \oplus E[p^r].$$

Po dokazanem zgoraj lahko zapišemo  $E[n'] \simeq \mathbb{Z}_{n'} \oplus \mathbb{Z}_{n'}$ , ker  $p \nmid n'$ . Vemo da velja zveza

$$\mathbb{Z}_{n'} \oplus \mathbb{Z}_{p^r} \simeq \mathbb{Z}_{n'p^r} \simeq \mathbb{Z}_n$$

. Od tod pa sledi

$$E[n] \simeq \mathbb{Z}_{n'} \oplus \mathbb{Z}_{n'} \text{ ali } \mathbb{Z}_n \oplus \mathbb{Z}_{n'}.$$

□

## 6 Weilovo parjenje

Parjenja eliptičnih krivulj so uporabna za konstrukcijo različnih kriptografskih sistemov, prav tako pa imajo pomembno vlogo pri napadih na problem diskretnega logaritma nad gladkimi kubičnimi krivuljami, katerimu bomo posvetili nadaljna poglavja.

### Definicija 6.1.

Naj bo  $K$  polje in naj bo  $n \in \mathbb{N}$  tak, da karakteristika  $K$  ne deli  $n$ .

$$\mu_n = \{x \in \overline{K} \mid x^n = 1\}$$

je grupa  $n$ -tih korenov enote grupe  $\overline{K}$ .

### Trditev 6.2.

Naj bo  $E$  eliptična krivulja definirana nad poljem  $K$ , in naj bo  $n \in \mathbb{N}$ . Predpostavimo, da karakteristika polja  $K$  ne deli  $n$ . Potem obstaja Weilovo parjenje

$$e_n : E[n] \times E[n] \rightarrow \mu_n,$$

za katerega velja:

- $e_n$  je bilinearna v obeh spremenljivkah

$$e_n(S_1 + S_2, T) = e_n(S_1, T)e_n(S_2, T)$$

in

$$e_n(S, T_1 + T_2) = e_n(S, T_1)e_n(S, T_2)$$

za vse  $S, S_1, S_2, T, T_1, T_2 \in E[n]$ .

- $e_n$  je ne degenerirana v obeh spremenljivkah. To pomeni če je  $e_n(S, T) = 1$  za vse  $T \in E[n]$  potem  $S = \infty$ , ter obratno.
- $e_n(T, T) = 1$  za vse  $T \in E[n]$
- $e_n(T, S) = e_n(S, T)^{-1}$  za vse  $S, T \in E[n]$
- $e_n(\rho S, \rho T) = \rho(e_n(S, T))$  za vse avtomorfizme  $\rho$  iz  $\overline{K}$ , za katere je  $\rho$  identiteta na koeficientih  $E$ .
- $e_n(\alpha(S), \alpha(T)) = e_n(S, T)^{\deg(\alpha)}$  za vse separabilne endomorfizme  $\alpha$  polja  $E$ .

### Posledica 6.3.

Naj bosta  $T_1, T_2$  baza  $E[n]$ . Potem je  $e_n(T_1, T_2)$  generator grupe  $\mu_n$ .

*Dokaz.*

Vemo, da za poljubni točki  $T_1, T_2$  velja  $e_n(T_1, T_2)^n = 1$ , ker se slika parjenja nahaja v grupi  $n$ -tih korenov enote. Pokazati moramo torej, da če za neko število  $d$  velja  $e_n(T_1, T_2)^d = 1$  potem od tod sledi, da je  $d \geq n$ . Recimo torej, da je  $e_n(T_1, T_2) = \zeta$ , kjer velja  $\zeta^d = 1$ . Po točki ena trditve 6.2 velja

$$e_n(T_1, dT_2) = e_n(T_1, T_2)^d = 1.$$

Prav tako velja  $e_n(T_2, dT_2) = e_n(T_2, T_2)^d = 1$ . Naj bo  $S \in E[n]$ , potem se  $S$  izraža kot  $S = aT_1 + bT_2$  za neka  $a, b \in \mathbb{N}$ . S ponovno uporabo trditve 6.2 vidimo, da velja

$$e_n(S, dT_2) = e_n(T_1, dT_2)^a e_n(T_2, dT_2)^b = 1.$$

Ker to velja za vsak  $S$  po točki dva trditve 6.2 sledi, da je  $dT_2 = \infty$ . To pa je mogoče le če  $n|d$ , kar pomeni da je  $n \leq d$ .  $\square$

**Posledica 6.4.**

*Če  $E[n] \subseteq E(K)$ , potem  $\mu_n \subset K$ .*

*Dokaz.*

Naj bo  $\rho$  tak avtomorfizem  $\overline{K}$ , da je  $\rho$  indentiteta na elementih  $K$ . Naj bosta  $T_1, T_2$  baza  $E[n]$ . Ker imata po predpostavki točki koordinate v  $K$  zanju velja  $\rho T_1 = T_1, \rho T_2 = T_2$ . Po trditvi 6.2 velja

$$\zeta = e_n(T_1, T_2) = e_n(\rho T_1, \rho T_2) = \rho(e_n(T_1, T_2)) = \rho(\zeta).$$

Eden od izrekov Galoisove teorije pa pravi, če je  $x \in \overline{K}$  fiksni za vse take avtomorfizme  $\rho$  potem je  $x \in K$ . Torej je  $\zeta \in K$ . Ker pa je  $\zeta$  primitivni koren enote od tod sledi  $\mu_n \subset K$ .  $\square$

Pred dokazom trditve 6.2 bomo potrebovali še eno trditev.

**Trditev 6.5** ([18], Trditev 9.34).

*Naj bo  $E$  eliptična krivulja na poljem  $K$ . Naj bo  $f$  funkcija, ki elementom iz  $E$  priredi element v  $\overline{K} \cup \{\infty\}$ , ter naj bo  $n \in \mathbb{N}$ , tako da  $n$  ni deljiv s karakteristiko  $K$ . Če velja  $f(P + T) = f(P)$  za vse  $P \in E(\overline{K}), T \in E[n]$ . Potem obstaja funkcija  $h$  na  $E$ , za katero velja*

$$f(P) = h(nP), \text{ za vse } P.$$

*Dokaz trditve 6.2.*

Naj bo  $T \in E[n]$ . Po izreku 4.10 obstaja funkcija  $f$ , tako da velja

$$\text{div}(f) = n[T] - n[\infty]. \quad (6.1)$$

To drži ker je  $D = n[T] - n[\infty]$  delitelj za katerega velja  $\deg(D) = 0$ . Prav tako pa velja  $\text{sum}(D) = \infty$ , ker se nahajamo v  $E[n]$ . Izberimo sedaj še točko  $T' \in E[n^2]$ , za katero velja  $nT' = T$ . Pokazati želimo, da obstaja funkcija  $g$ , tako da velja

$$\text{div}(g) = \sum_{R \in E[n]} ([T' + R] - [R]).$$

Preden lahko uporabimo izrek 4.10 moramo preveriti, da vse potrebne lastnosti držijo. Veljati mora torej

$$\text{sum}\left(\sum_{R \in E[n]} ([T' + R] - [R])\right) = \infty$$

in

$$\deg\left(\sum_{R \in E[n]} ([T' + R] - [R])\right) = 0.$$

Po izreku 5.2, se da  $E[n]$  zapisati kot  $\mathbb{Z}_n \oplus \mathbb{Z}_n$ . To pomeni, da  $E[n]$  vsebuje  $n^2$  različnih točk  $R$ . Velja torej

$$\text{sum}\left(\sum_{R \in E[n]} ([T' + R] - [R])\right) = \sum_{R \in E[n]} T' + R - R = \sum_{R \in E[n]} T' = n^2 T' = nT = \infty.$$

Podobno preverimo, še da velja  $\deg(\sum_{R \in E[n]} ([T' + R] - [R])) = 0$ . Torej po izreku 4.10 taka funkcija  $g$  obstaja. Označimo z  $f \circ n$  funkcijo, ki začne z neko točko, jo pomnoži z  $n$  in nato na njej uporabi  $f$ . Točke  $P = T' + R$ , kjer je  $R \in E[n]$  so točke za katere velja  $nP = T$ . Iz 6.1 sledi

$$\text{div}(f \circ n) = n\left(\sum_{R \in E[n]} [T' + R]\right) - n\left(\sum_{R \in E[n]} [R]\right) = \text{div}(g^n).$$

Iz definicije delitelja funkcije sledi, da je funkcija  $f \circ n$  oblike  $g^n \times$  neka konstanta. Če za nov  $f$  vzamemo ta  $f$  pomnožen s primerno konstanto, potem lahko predpostavimo, da velja

$$f \circ n = g^n.$$

Naj bo  $S \in E[n]$ , ter naj bo  $P \in E(\overline{K})$ . Potem velja

$$g(P + S)^n = f(n(P + S)) = f(nP) = g(P)^n.$$

Zaradi tega velja  $g(P + S)/g(P) \in \mu_n$ . Tako lahko sedaj definiramo Weilovo parjenje kot

$$e_n(S, T) = \frac{g(P + S)}{g(P)}.$$

Ta definicija je dobra, ker je  $g$  zaradi delitelja določen do skalarja natančno, zaradi tega pa je definicija neodvisna od izbire  $g$ . Prav tako pa je definicija neodvisna od izbire  $P$ , a je obrazložitev bolj zahtevna in je ne bomo navedli. Lahko jo najdemo v [15]

Sedaj, ko smo uspeli definirati Weilovo parjenje moramo preveriti, da zanj veljajo lastnosti 1-6.

1. Ker je definicija neodvisna od izbire točke  $P$  lahko uporabimo  $P, P + S_1$ .

$$\begin{aligned} e_n(S_1, T)e_n(S_2, T) &= \frac{g(P + S_1)}{g(P)} \frac{g(P + S_1 + S_2)}{g(P + S_1)} \\ &= \frac{g(P + S_1 + S_2)}{g(P)} \\ &= e_n(S_1 + S_2, T). \end{aligned}$$

Pokazati moramo še

$$e_n(S, T_1)e_n(S, T_2) = e_n(S, T_1 + T_2).$$

Dokaz linearnosti v drugi spremenljivki pa je malce težji kot dokaz za prvo spremenljivko. Predpostavimo da  $T_1, T_2, T_3 \in E[n]$  z lastnostjo  $T_1 + T_2 = T_3$ . Označimo z  $f_i, g_i$  funkcije ki spadajo k definicijam  $e_n(S, T_i)$ . Po izreku 4.10 obstaja funkcija  $h$ , za katero velja

$$\operatorname{div}(h) = [T_3] - [T_1] - [T_2] + [\infty].$$

Enačba 6.1 nam da

$$\operatorname{div}\left(\frac{f_3}{f_1 f_2}\right) = n \operatorname{div}(h) = \operatorname{div}(h^n).$$

Zato obstaja konstanta  $c \in \overline{K}^\times$ , tako da velja

$$f_3 = c f_1 f_2 h^n.$$

Od tod pa sledi

$$g_3 = c^{1/n}(g_1)(g_2)(h \circ n)$$

.

To pa nas končno pripelje do

$$\begin{aligned} e_n(S, T_1 + T_2) &= \frac{g_3(P + S)}{g_3(P)} = \frac{g_1(P + S)}{g_1(P)} \frac{g_2(P + S)}{g_2(P)} \frac{h(n(P + S))}{h(nP)} \\ &= e_n(S, T_1) e_n(S, T_2). \end{aligned}$$

Tu smo upoštevali  $nS = \infty$ , od koder sledi  $h(n(P + S)) = h(nP)$ .

2. Predpostavimo, da  $T \in E[n]$ , tak da velja  $e_n(S, T) = 1$  za vse  $S \in E[n]$ . To pomeni, da je  $g(P + S) = g(P)$  za vse  $P$  in  $S \in E[n]$ . Po trditvi 6.5 obstaja funkcija  $h$ , tako da velja  $g = h \circ [n]$ . Od tod sledi

$$(h \circ n)^n = g^n = f \circ n.$$

Ker je množenje z  $n$  surjektivno na  $E(\overline{K})$ , od tod sledi  $h^n = f$ . To pa pomeni

$$n \operatorname{div}(h) = \operatorname{div}(f) = n[T] - n[\infty],$$

kar pa pomeni, da je  $\operatorname{div}(h) = [T] - [\infty]$ . Po izreku 4.10 pa od tod sledi  $T = \infty$ . S tem smo, dokazali eno polovico točke 2, druga polovica pa avtomatično sledi iz tega ter uporabe točke 4.

3. Označimo z  $\tau_{jT}$  točko  $jT$ . V tem primeru  $f \circ \tau_{jT}$  označuje funkcijo  $P \mapsto f(P + jT)$ . Delitelj te funkcije je torej  $n[T - jT] - n[-jT]$ . Zaradi tega velja

$$\begin{aligned} \operatorname{div}\left(\prod_{j=0}^{n-1} f \circ \tau_{jT}\right) &= \sum_{j=0}^{n-1} (n[(1-j)T] - n[-jT]) \\ &= n[T] - n[-T] + n[-T] - n[-2T] + \cdots + n[(-n+2)T] - n[(-n+1)T] \\ &= n[T] - n[(-n+1)T] = n[T] - n[-nT + T] = n[T] - n[\infty + T] \\ &= n[T] - n[T] = 0 \end{aligned}$$

To pomeni, da je funkcija  $\prod_{j=0}^{n-1} f \circ \tau_{jT}$  konstantna.  $N$ -ta potenca funkcije  $\prod_{j=0}^{n-1} g \circ \tau_{jT'}$  pa je ravno produkt  $f$  komponirane z  $n$ .

$$\begin{aligned} \left( \prod_{j=0}^{n-1} g \circ \tau_{jT'} \right)^n &= \prod_{j=0}^{n-1} f \circ n \circ \tau_{jT'} \\ &= \prod_{j=0}^{n-1} f \circ \tau_{jT} \end{aligned}$$

To pa tudi pomeni, da je funkcija  $\prod_{j=0}^{n-1} g \circ \tau_{jT'}$  konstantna. To pa pomeni, da lahko namesto točke  $P$  vanjo vstavimo točko  $P + T'$  in dobimo

$$\prod_{j=0}^{n-1} g(P + T' + jT') = \prod_{j=0}^{n-1} g(P + jT').$$

Ko pokrajšamo stvari na levi in desni strani, dobimo

$$g(P + nT') = g(P).$$

Ker pa velja  $nT' = T$ , to ravno pomeni

$$e_n(T, T) = \frac{g(P + T)}{g(P)} = 1.$$

4. Iz točke 1 in 3 sledi

$$\begin{aligned} 1 &= e_n(S + T, S + T) = e_n(S, S)e_n(S, T)e_n(T, S)e_n(T, T) \\ &= e_n(S, T)e_n(T, S). \end{aligned}$$

To pa pomeni  $e_n(T, S) = e_n(S, T)^{-1}$ .

5. Naj bo  $\rho$  avtomorfizem na  $\overline{K}$ , za katerega velja, da je  $\rho$  identiteta na koeficientih  $E$ . Če  $\rho$  apliciramo na vsakem koraku konstrukcije Weilovega parjenja dobimo

$$\operatorname{div}(f^\rho) = n[\rho T] - n[\infty],$$

ter podobno za  $g^\rho$ . Tu  $f^\rho, g^\rho$  označujeta funkciji, ki jih dobimo tako, da  $\rho$  apliciramo na koeficiente racionalnih funkcij, ki definirajo  $f, g$ . Zato velja

$$\rho(e_n(S, T)) = \rho\left(\frac{g(P + S)}{g(P)}\right) = \frac{g^\rho(\rho P + \rho S)}{g^\rho(\rho P)} = e_n(\rho S, \rho T).$$

6. Naj bo  $\{Q_1, \dots, Q_k\} = \operatorname{Ker}(\alpha)$ . Ker je  $\alpha$  separabilen, po izreku 3.25 velja  $k = \deg(\alpha)$ . Naj bo

$$\operatorname{div}(f_T) = n[T] - n[\infty], \quad \operatorname{div}(f_{\alpha(T)}) = n[\alpha(T)] - n[\infty]$$

in

$$g_T^n = f_T \circ n, \quad g_{\alpha(T)}^n = f_{\alpha(T)} \circ n.$$

Če uporabimo oznako  $\tau_Q$  iz točke 3 velja

$$\operatorname{div}(f_t \circ \tau_{-Q_i}) = n[T + Q_i] - n[Q_i].$$

Zato velja

$$\begin{aligned} \operatorname{div}(f_{\alpha(T)} \circ \alpha) &= n \sum_{T'', \alpha(T'')=\alpha(T)} [T''] - n \sum_{Q, \alpha(Q)=\infty} [Q] \\ &= n \sum_i ([T + Q_i] - [Q_i]) \\ &= \operatorname{div}\left(\prod_i (f_T \circ \tau_{-Q_i})\right) \end{aligned}$$

Za vsak  $i$  sedaj izberemo  $Q'_i$ , tako da velja  $nQ'_i = Q_i$ . Potem

$$g_T(P - Q'_i)^n = f_T(nP - Q_i).$$

Za to funkcijo velja

$$\begin{aligned} \operatorname{div}\left(\prod_i (g_T \circ \tau_{-Q'_i})^n\right) &= \operatorname{div}\left(\prod_i f_T \circ \tau_{-Q_i} \circ n\right) \\ &= \operatorname{div}(f_{\alpha(T)} \circ \alpha \circ n) \\ &= \operatorname{div}(f_{\alpha(T)} \circ n \circ \alpha) \\ &= \operatorname{div}(g_{\alpha(T)} \circ \alpha)^n. \end{aligned}$$

Delitelja  $\prod_i g_T \circ \tau_{-Q'_i}$  in  $g_{\alpha(T)} \circ \alpha$  se torej razlikujeta samo za produkt neke konstante  $C$ .

Velja

$$\begin{aligned} e_n(\alpha(S), \alpha(T)) &= \frac{g_{\alpha(T)}(\alpha(P + S))}{g_{\alpha(T)}(\alpha(P))} \\ &= \prod_i \frac{g_T(P + S - Q'_i)}{g_T(P - Q'_i)} \\ &= \prod_i e_n(S, T) \\ &= e_n(S, T)^k = e_n(S, T)^{\deg(\alpha)}. \end{aligned}$$

□

## 6.1 Učinkovit algoritem za izračun Weilovega parjenja

Leta 1986 je Vicor Miller napisal članek o tem, kako učinkovito izračunati Weilovo parjenje [13].

### Izrek 6.6.

Naj bo  $E$  eliptična krivulja in naj bosta  $P = (x_P, y_P), Q = (x_Q, y_Q)$  ne ničelni točki na  $E$ .

1. Označimo z  $\lambda$  naklon premice, ki povezuje točki  $P, Q$ . V primeru, da sta ti točki enaki,  $\lambda$  predstavlja naklon tangente v točki. Če je premica navpična ( $x_P = x_Q$ ), potem privzamemo, da je  $\lambda = \infty$ . Definirajmo funkcijo  $g_{P,Q}$  na sledeči način:

$$g_{P,Q} = \begin{cases} \frac{y - y_P - \lambda(x - x_P)}{x + x_P + x_Q - \lambda^2} & \text{če } \lambda \neq \infty, \\ x - x_P & \text{sicer.} \end{cases}$$

Potem velja

$$\text{div}(g_{P,Q}) = [P] + [Q] - [P + Q] - [\infty].$$

2. (Millerjev algoritem) Naj bo  $m \geq 1$ . Zapišimo  $m$  v binarnem kot

$$m = m_0 + m_1 \cdot 2 + m_2 \cdot 2^2 + \dots + m_{n-1} \cdot 2^{n-1},$$

kjer so  $m_i \in \{0, 1\}$  in  $m_{n-1} \neq 0$ . Potem algoritem 2 vrne funkcijo  $f_P$ , za katero velja

$$\text{div}(f_P) = m[P] - [mP] - (m - 1)[\infty].$$

---

### Algoritem 2 Millerjev algoritem

---

Vhod: število  $m$  podano v binarnem zapisu, točka  $P$  na eliptični krivulji

Izhod: funkcija  $f_P$

$T = P, f = 1$

**for**  $i = n - 2 : 0$  **do**

$f = f^2 \cdot g_{T,T}$

$T = 2T$

**if**  $m_i = 1$  **then**

$f = f \cdot g_{T,P}$

$T = T + P$

**end if**

**end for**

---

*Dokaz.*

1. Predpostavimo najprej, da  $\lambda \neq \infty$ , ter naj  $y = \lambda x + \mu$  predstavlja premico skozi  $P, Q$  (ali tangento v primeru, da je  $P = Q$ ). Taka premica seka krivuljo  $E$  v treh točkah  $P, Q, -P - Q$ . To sledi iz sestave grupe nad  $E$ . Za to premico torej velja

$$\text{div}(y - \lambda x - \mu) = [P] + [Q] + [-P - Q] - 3[\infty].$$



Navpične premice pa sekajo  $E$  neki točki  $P$  in  $-P$ . Torej velja

$$\operatorname{div}(x - x_{P+Q}) = [P + Q] + [-P - Q] - 2[\infty].$$

Od tod sledi, da ima funkcija

$$g_{P,Q} = \frac{y - \lambda x - \mu}{x - x_{P+Q}}$$

željen delitelj

$$\operatorname{div}(g_{P,Q}) = [P] + [Q] - [P + Q] - [\infty].$$

Z uporabo formule za seštevanje točk sedaj lahko to funkcijo preoblikujemo v želeno obliko

$$g_{P,Q} = \frac{y - y_P - \lambda(x - x_P)}{x + x_P + x_Q - \lambda^2}.$$

V primeru da velja  $\lambda = \infty$ , potem velja  $P + Q = \infty$ . V tem primeru hočemo imeti delitelj oblike  $\operatorname{div}(g_{P,Q}) = [P] + [-P] - 2[\infty]$ . Tak delitelj pa ima ravno funkcija  $x - x_P$ .

2. Dokažemo s pomočjo indukcije, pri čemer upoštevamo  $\operatorname{div}(g_{T,T}) = 2[T] - [2T] - [\infty]$ ,  $\operatorname{div}(g_{T,P}) = [T] + [P] - [T + P] - [\infty]$ . Preverimo, na primeru  $m = 3$ . Binarni zapis  $m$  je torej 11. Sledimo korakom algoritma 2.

$$f = f^2 \cdot g_{T,T} = g_{P,P}$$

Ker je v tem primeru  $m_0 = 1$  moramo  $f$  popraviti v  $f = f \cdot g_{T,P}$ . To nam da

$$f = g_{P,P} \cdot g_{2T,P}.$$

Če uporabimo sedaj zgornji zvezi dobimo

$$\operatorname{div}(f) = 2[P] - [2P] - [\infty] + [2P] + [P] - [2P + P] - [\infty] = 3[P] - [3P] - 2[\infty].$$

To pa je ravno to kar želimo.

□

S pomočjo izreka 6.6 lahko sedaj izračunamo Weilovo parjenje  $e_m(P, Q)$  kot

$$e_m(P, Q) = \frac{\frac{f_P(Q+S)}{f_P(S)}}{\frac{f_Q(P-S)}{f_Q(-S)}}.$$

Tu moramo za točko  $S$  izbrati  $S \notin \{\infty, P, -Q, P - Q\}$ .

### Primer 6.7.

Naj bo  $y^2 = x^3 + 30x + 34$  eliptična krivulja nad poljem  $\mathbb{F}_{631}$ . Izberimo točki  $P = (36, 60)$ ,  $Q = (121, 387)$ . Izračunati želimo Weilovo parjenje  $e_5(P, Q)$

Če želimo uporabiti Millerjev algoritem moramo izbrati točko  $S$ . Izberimo  $S = (0, 36)$ . Hitro lahko preverimo, da  $S$  res zadošča kriterijem izbire. Za izračun

$e_5(P, Q)$  moramo izračunati  $f_P, f_Q$  v dveh različnih točkah. Poglejmo si podrobnejši izračun za  $f_P(S)$ .

V prvem koraku, moramo izračunati  $g_{P,P}(S)$ . Po vseh potrebnih izračunih dobimo

$$\text{naklon} = 569, \text{ števec} = 268, \text{ imenovalec} = 14.$$

Od tod torej sledi  $g_{P,P}(S) = 560$ . Ker velja  $m_1 = 0$ . Torej po prvem koraku dobimo  $f = 560, T = 2P$ .

V drugem koraku moramo poračunati  $g_{2P,2P}(S)$ , kjer kot rezultat dobimo 399. Vrednost  $f$  pa moramo popraviti na 362. V tem koraku velja  $m_0 = 1$ , torej moramo tu poračunati še  $g_{4P,P}(S)$ . Premica med  $P, 4P$  je navpična zato uporabimo drugo vejo funkcije  $g$ . Kot rezultat dobimo  $g_{4P,P}(S) = 595$ . Še zadnjič popravimo  $f$  in kot končni rezultat dobimo  $f_P(S) = 219$ .

Na podoben način sedaj poračunamo še ostale vrednosti, ter dobimo

$$\frac{f_P(Q+S)}{f_P(S)} = \frac{103}{219} = 473,$$

$$\frac{f_Q(P-S)}{f_Q(-S)} = \frac{284}{204} = 88.$$

Kot končni rezultat imamo torej

$$e_5(P, Q) = \frac{473}{88} = 242.$$

## 6.2 Implementacija Millerjevega algoritma

Tu je podana implementacija Millerjevega algoritma v programskem okolju Python.

```
from base import *
```

```
def naklon(P,Q):  
    """
```

*Opis:*

*Funkcija naklon izracuna naklon premice med točkama  
P in Q, ki lezita na elipticni krivulji*

*Definicija:*

*naklon(P,Q)*

*Vhodni podatki:*

*P... razred Point(EllipticCurve), ki predstavlja točko na*

*elipticni krivulji in je definiran v dodatku base*

*Q... razred Point(EllipticCurve), ki predstavlja točko na*

*elipticni krivulji in je definiran v dodatku base*

*Izhodni podatek:*

*stevilo po modulu P.mod, ki predstavlja naklon*

```

"""
mod = P.mod
if P == Q:
    st = (3*(P.x**2)+P.a) % mod
    im = NumberMod(2*(P.y),mod).inverse().num
    rez = (st*im) % mod
    return rez
else:
    st = (Q.y-P.y) % mod
    im = NumberMod(Q.x-P.x,P.mod).inverse().num
    rez = (st*im) % mod
    return rez

def g(P,Q,X):
    """
    Opis:
        Funkcija g del funkcije potrebne za izracun Weilovega
        parjenja, s pomocjo Millerjevega algoritma.
        Funkcija izracuna funkcijo  $g_{\{P,Q\}}(X)$ 

    Definicija:
         $g(P,Q,X)$ 

    Vhodni podatki:
        P... razred Point(EllipticCurve), ki predstavlja točko na
        elipticni krivulji in je definiran v dodatku base
        Q... razred Point(EllipticCurve), ki predstavlja točko na
        elipticni krivulji in je definiran v dodatku base
        X... razred Point(EllipticCurve), ki predstavlja točko na
        elipticni krivulji in je definiran v dodatku base

    Izhodni podatek:
        stevilo, ki predstavlja vrednost  $g_{\{P,Q\}}(X)$ 
    """
    if P.x == Q.x and P.y != Q.y:
        rez = (X.x-P.x) % P.mod
        return rez
    else:
        #lambda je naklon premice
        lam = naklon(P,Q)
        st = (X.y-P.y-lam*(X.x-P.x)) % P.mod
        im = NumberMod(X.x+P.x+Q.x-lam**2,P.mod)
        im = im.inverse().num
        rez = (st*im) % P.mod

```

```

    return rez

def Miller(P,X,m):
    """
    Opis:
        Funkcija Miller je implementacija Millerjevega
        algoritma potrebnega za izracun Weilovega
        parjenja. Ce je
        $$e_m(P,Q)=(f_P(Q+S)/f_P(S))/(f_Q(P-S)/f_Q(-S))$$
        potem funkcija Miller predstavlja izracun
        vrednosti $f_P(X)$.

    Definicija:
        Miller(P,X,m)

    Vhodni podatki:
        P... razred Point, ki predstavlja tocko na
            elipticni krivulji
        X... razred Point, ki predstavlja tocko na
            elipticni krivulji
        m... red tocke P

    Izhodni podatek:
        vrednost funkcije $f_P(X)$
    """
    binarno = bin(m)[2:]
    #vrne niz porezemo 0b na zacetku niza
    n = len(binarno)
    T = P
    f = 1
    for i in range(1,n):
        f = (f**2 * g(T,T,X)) % P.mod
        T = 2*T
        if int(binarno[i]) == 1:
            f = (f* g(T,P,X)) % P.mod
            T = T + P
    return f

def WeilPairing(P,Q,S,N):
    """
    Opis:
        Funkcija WeilPairing je implementacija Weilovega
        parjenja $e_N(P,Q)$, kjer je
        $$e_N(P,Q)=(f_P(Q+S)/f_P(S))/(f_Q(P-S)/f_Q(-S))$$

```

*Definicija:*

*WeilPairing(P,Q,S,N)*

*Vhodni podatki:*

*P... razred Point(EllipticCurve), ki predstavlja točko na elipticni krivulji in je definiran v dodatku base*

*Q... razred Point(EllipticCurve), ki predstavlja točko na elipticni krivulji in je definiran v dodatku base*

*S... razred Point(EllipticCurve), ki predstavlja točko, ki ni v podgrupi generirani z P,Q*

*N... večkratnik reda točke P*

*Izhodni podatek:*

*int eN, ki predstavlja vrednost Weilovega parjenja*  
"""

*#test ali je S v podgrupi generirani z P,Q*

fpQS = Miller(P,Q+S,N)

fpS = Miller(P,S,N)

fqPS = Miller(Q,P-S,N)

fqS = Miller(Q,-S,N)

fpS = NumberMod(fpS,P.mod).inverse().num

eN1 = (fpQS \* fpS) % P.mod

fqS = NumberMod(fqS,P.mod).inverse().num

eN2 = (fqPS \* fqS) % P.mod

eN2 = NumberMod(eN2,P.mod).inverse().num

eN = (eN1\*eN2) % P.mod

**return eN**

E = EllipticCurve(30,34,631)

P = Point(30,34,631,36,60)

Q = Point(30,34,631,121,387)

S = E.rand()

eN = WeilPairing(P,Q,S,5)

## 7 Problem diskretnega logaritma

Diffie-Hellmanova izmenjava ključev je postopek, pri katerem se dve osebi npr. Alenka in Boris dogovorita za skrivni ključ. To naredita tako, da tudi v primeru ko njun pogovor posluša tretji nepovabljeni gost npr. Ciril, le ta iz pogovora ne more rekonstruirati ključa, za katerega sta se tekom pogovora dogovorila Alenka in Boris.

---

**Algoritem 3** Diffie-Hellmanova izmenjava ključev.

---

1. Alenka in Boris se dogovorita za eliptično krivuljo  $E$  nad končnim obsegom  $\mathbb{F}_q$ , ter za točko  $P \in E(\mathbb{F}_q)$ .
  2. Alenka se naključno odloči za skrivno število  $a \in \mathbb{N}$ , in izračuna  $P_a = aP$ , ter to pošlje Borisu. Pri tem red točke  $P$  ne sme biti enak  $a$ .
  3. Boris se naključno odloči za skrivno število  $b \in \mathbb{N}$ , in izračuna  $P_b = bP$ , ter to pošlje Alenki. Pri tem red točke  $P$  ne sme biti enak  $b$ .
  4. Alenka izračuna  $aP_b = abP$ .
  5. Boris izračuna  $bP_a = baP$ .
  6. Skupni ključ je  $abP$ .
- 

Kot sam ključ bi lahko na koncu Alenka in Boris uporabila npr. zadnjih 256 bitov x-koordinate točke  $abP$ . Tu se zanašamo na to, da je iz  $E, \mathbb{F}_q, P, P_a, P_b$  težko izračunati  $baP$ . Zelo veliko pa je tu odvisno od same izbire krivulje  $E$ .

To nas privede do t. i. problema diskretnega logaritma v grupi točk nad eliptičnimi krivuljami.

**Definicija 7.1.** Naj bo  $G$  grupa, kjer njeno operacijo označimo z  $\circ$ . Naj bosta  $a, b \in G$ . Naj  $b^k$  označuje

$$b^k = \underbrace{b \circ b \circ \dots \circ b}_{k\text{-krat}}.$$

Število  $k \in \mathbb{N}$ , ki reši enačbo

$$b^k = a$$

imenujemo *diskretni logaritem* elementa  $a$  pri osnovi  $b$ .

Problem diskretnega logaritma je torej poiskati tak  $k$ , ki bo rešil dano enačbo.

**Trditev 7.2.**

*Če lahko rešimo problem diskretnega logaritma, potem smo rešili tudi problem Diffie-Hellmanove izmenjave ključev. Povedano drugače velja*

$$DL \Rightarrow DH.$$

*Dokaz.*

Problem Diffie-Hellmanove izmenjave ključev lahko enostavno prevedemo na problem diskretnega logaritma na sledeč način:

- Vzemi  $aP$  in izračunaj  $a$  tako, da rešiš problem diskretnega logaritma.
- Izračunaj  $a(bP)$ .

□

Na podoben način lahko definiramo tudi odločitveni Diffie-Hellmanov problem, ki se glasi.

**Definicija 7.3.**

Naj bodo dani  $P, aP, bP, Q$  in  $E(F_q)$ . Ali velja  $Q = abP$ ?

Povedano drugače, če dobimo namig ki vsebuje  $abP$ , ali lahko preverimo če ta informacija drži.

Poglejmo si na primeru, kako lahko s pomočjo Weilovega parjenja, v določenih primerih, pridemo do rešitve tega problema.

**Primer 7.4.**

Naj bo  $E$  podana kot  $y^2 = x^3 + 1$  nad  $\mathbb{F}_q$ , kjer je  $q \equiv 2 \pmod{3}$ . Naj bo  $\omega \in \mathbb{F}_{q^2}$  primitivni tretji koren enote. Opazimo, da  $\omega \notin \mathbb{F}_q$ , saj je red  $\mathbb{F}_q^\times$  enak  $q - 1$ , kar pa ni deljivo s tri. Definirajmo preslikavo

$$\beta : E(\overline{\mathbb{F}_q}) \rightarrow E(\overline{\mathbb{F}_q}), (x, y) \mapsto (\omega x, y), \beta(\infty) = \infty.$$

Preverimo lahko, da je tako podana preslikava izomorfizem. Denimo, da ima  $P \in E(\overline{\mathbb{F}_q})$  red  $n$ . Potem iz lastnosti izomorfizmov sledi, da ima tudi  $\beta(P)$  red  $n$ . Definirajmo modificirano Weilovo parjenje

$$e'_n(P_1, P_2) = e_n(P_1, \beta(P_2)),$$

kjer je  $e_n$  običajno Weilovo parjenje in  $P_1, P_2 \in E[n]$ .

Predpostavimo sedaj, da poznamo  $P, aP, bP, Q$  in želimo preveriti ali velja  $Q = abP$ . Najprej preverimo, če je  $Q$  večkratnik  $P$ . Po trditvi 8.1 bo to res natanko tedaj ko  $e_n(P, Q) = 1$ . Predpostavimo, da velja  $Q = tP$ , za nek  $t \in \mathbb{N}$ . Potem imamo

$$e'_n(aP, bP) = e'_n(P, P)^{ab} = e'_n(P, abP),$$

ter

$$e'_n(Q, P) = e'_n(P, P)^t.$$

Če predpostavimo  $3 \nmid n$ , potem je  $e'_n(P, P)$   $n$ -ti koren enote. To pa pomeni

$$Q = abP \iff t \equiv ab \pmod{n} \iff e'_n(aP, bP) = e'_n(P, P)^t.$$

V zgornjem primeru je potrebna še dodatna utemeljitev, ki jo lahko zapišemo kot lemo.

**Lema 7.5.**

Naj  $3 \nmid n$ . Če ima  $P \in E(\mathbb{F}_q)$  red  $n$ , potem je  $e'_n(P, P)$   $n$ -ti koren enote.

*Dokaz.*

Naj bo  $uP = v\beta(P)$  za neki števili  $u, v$ . Potem zaradi lastnosti izomorfizmov velja

$$\beta(vP) = v\beta(P) = uP \in E(\mathbb{F}_q).$$

Obravnavajmo sedaj dva primera.

- Če je  $vP = \infty$ , potem po definiciji  $\beta$  sledi  $uP = \infty$ . To pa pomeni, da je  $u \equiv 0 \pmod{n}$ .
- Če velja  $vP \neq \infty$ , potem zapišimo  $vP = (x, y)$ , kjer sta  $x, y \in \mathbb{F}_q$ . Velja

$$(\omega x, y) = \beta(vP) \in E(\mathbb{F}_q).$$

Ker  $\omega \notin \mathbb{F}_q$ , mora veljati  $x = 0$ . Torej je točka  $vP$  oblike  $(0, \pm 1)$ . Take točke pa imajo red tri, kar lahko poračunamo z definicijo grupe. To pa ni možno saj smo predpostavili  $3 \nmid n$ .

V obeh primerih mora torej veljati  $u, v \equiv 0 \pmod{n}$ . Od tod pa sledi, da sta  $P$  in  $\beta(P)$  baza  $E[n]$ . Po posledici 6.3 to pomeni, da je  $e'_n(P, P)$   $n$ -ti koren enote.  $\square$

## 7.1 Izračun indeksa

Izračun indeksa je postopek s pomočjo katerega lahko rešimo problem diskretnega logaritma v polju  $\mathbb{F}_p$ . Pri metodi si najprej izberemo množico majhnih praštevil, ki jo poimenujemo baza faktorizacije. Označimo to množico z  $B = \{p_1, p_2, \dots, p_B\}$ . Prvi korak izračuna indeksa je, izračunati diskretne logaritme vseh elementov baze. V drugem koraku pa nato s pomočjo baze izračunamo diskretni logaritem za poljuben element.

Poglejmo si sedaj podrobneje prvi korak tega algoritma. Da poračunamo diskretne logaritme baze pri osnovi  $\alpha$  najprej sestavimo vsaj  $B$  enačb oblike

$$\alpha^{x_j} \equiv p_1^{a_{1j}} \dots p_B^{a_{Bj}} \pmod{p},$$

$1 \leq j \leq B$ . Te enačbe so ekvivalentne enačbam oblike

$$x_j \equiv a_{1j} \log_\alpha p_1 + \dots + a_{Bj} \log_\alpha p_B \pmod{p-1}.$$

Če smo izbrali enačbe primerno, potem bo imel sistem enolično rešitev modulo  $p-1$ . To pa ravno pomeni, da smo našli diskretne logaritme baze. Vprašanje je torej še kako generiramo te enačbe. Eden iz med načinov je, da izberemo naključno število  $x$ , ter poračunamo  $\alpha^x \pmod{p}$ . Za tem pa pogledamo, če ima dobljeno število vse faktorje v bazi.

Zaradi drugega koraka v algoritmu, lahko uvrstimo izračun indeksa pod algoritme tipa Las Vegas, saj naključno izberemo število  $1 \leq s \leq p-2$ , ter izračunamo  $\gamma \equiv \beta\alpha^s \pmod{p}$ . V tej enačbi je  $\beta$  število, katerega diskretni logaritem nas zanima. Sedaj pa  $\gamma$  probamo zapisati kot produkt faktorjev iz baze. Denimo da lahko zapišemo

$$\beta\alpha^s \equiv p_1^{c_1} \dots p_B^{c_B} \pmod{p}.$$



To lahko prepišemo v obliko

$$\log_{\alpha} \beta + s \equiv c_1 \log_{\alpha} p_1 + \dots + c_B \log_{\alpha} p_B \pmod{p-1}.$$

Tu pa poznamo vse spremenljivke razen  $\log_{\alpha} \beta$ .

Poglejmo si sedaj algoritem še na primeru.

**Primer 7.6.**

Naj bo  $p = 1217$  praštevilo in naj bo  $g = 3$  generator ciklične grupe  $\mathbb{F}_p^{\times}$ . Za  $h = 37$  želimo rešiti problem diskretnega logaritma. Naj  $L(x)$  označuje diskretni logaritem  $x$  pri osnovi  $b$ , torej

$$g^{L(x)} \equiv x \pmod{p}.$$

Izberimo si bazo praštevil  $\{2, 3, 5, 7, 11, 13\}$ . Pri tem upoštevamo, da bo večja baza pomenila več računanja, a hkrati lažjo pot do odgovora. Iščemo  $x$ -e tako, da bo

$$3^x \equiv \pm \text{produktu praštevil iz baze} \pmod{1217}.$$

Ob iskanju takih  $x$  najdemo naslednje enakosti:

$$\begin{aligned} 3^1 &\equiv 3 \pmod{1217} \\ 3^{24} &\equiv -2^2 \cdot 7 \cdot 13 \pmod{1217} \\ 3^{25} &\equiv 5^3 \pmod{1217} \\ 3^{30} &\equiv -2 \cdot 5^2 \pmod{1217} \\ 3^{54} &\equiv -5 \cdot 11 \pmod{1217} \\ 3^{87} &\equiv 13 \pmod{1217} \end{aligned}$$

Z večjo bazo bi v tem primeru lažje našli take enačbe, a bi jih hkrati potrebovali več. Z uporabo malega Fermatovega izreka, velja

$$3^{1216} \equiv 1 \equiv (-1)^2 \pmod{1217},$$

od koder sledi  $L(-1) \equiv 608 \pmod{1216}$ . Če enačbe sedaj zapišemo z uporabo  $L(h)$ , dobimo

$$\begin{aligned} 1 &\equiv L(3) \pmod{1216} \\ 24 &\equiv 608 + 2L(2) + L(7) + L(13) \pmod{1216} \\ 25 &\equiv 3L(5) \pmod{1216} \\ 30 &\equiv 608 + L(2) + 2L(5) \pmod{1216} \\ 54 &\equiv 608 + L(5) + L(11) \pmod{1216} \\ 87 &\equiv L(13) \pmod{1216} \end{aligned}$$

Od tod dobimo  $L(2) = 216, L(11) = 1059, L(7) = 113, L(5) = 819, L(13) = 87, L(3) = 1$ . Sedaj poračunamo za različne  $j$  vrednost  $3^j * 37$ , dokler ne dobimo  $3^j * 37 \equiv \text{produktu elementov iz baze}$ . Pri vrednosti  $j = 16$  dobimo

$$3^{16} \cdot 37 \equiv 2^3 \cdot 7 \cdot 11 \pmod{1217}.$$

Iščemo  $L(37)$ , iz definicije  $L$  pa velja

$$3^{L(37)} \equiv 37 \pmod{1217} \equiv 2^3 \cdot 7 \cdot 11 \cdot 3^{-16} \pmod{1217}.$$

Če sedaj namesto baze vstavimo primerne  $L$  dobimo

$$3^{L(37)} \equiv 3^{3L(2)} \cdot 3^{L(7)} \cdot 3^{L(11)} \cdot 3^{-16L(3)} \pmod{1217}.$$

$L(37)$  lahko sedaj zapišemo kot

$$L(37) \equiv 3L(2) + L(7) + L(11) - 16L(3) \pmod{1216} \equiv 588 \pmod{1216}.$$

Torej je naš iskani  $k = 588$ .

Časovna zahtevnost algoritma je  $O(e^{(1+o(1))\sqrt{\ln p \ln \ln p}})$  [16].

## 7.2 Izračun indeksa program

Nadaljne programe bomo pisali v prosto dostopnem programskem okolju SAGE. S pomočjo tega si bomo prihranili ogromno dela pri implementaciji različnih matematičnih objektov, ki jih bomo potrebovali. Z uporabo programskega okolja SAGE, lahko napad enostavno tudi sprogramiramo.

**def** IndexCalculus(g, h, q, baza):

"""

*Opis:*

*IndexCalculus vrne tak  $k$ , da velja  $g^k \equiv h \pmod{q}$ , kjer je  $g$  generator multiplikativne grupe  $\mathbb{Z}_q^*$ .*

*Zgled:*

$g = 3$   
 $q = 1217$   
 $h = 37$   
 $baza = [-1, 2, 3, 5, 7, 11, 13]$

*Definicija:*

*IndexCalculus(g, h, q, baza)*

*Vhodni podatki:*

*g... generator multiplikativne grupe  $\mathbb{Z}_q^*$ .*  
*q... modul s katerim delamo*  
*h... desna stran problema, ki ga resujemo*  
*baza... baza prastevil s katerimi delamo more vsebovati tudi -1*

*Izhodni podatek:*

*$k$ , da velja  $g^k \equiv h \pmod{q}$*   
 """

```

seznam_relacij = []
V = VectorSpace(GF(q), len(baza)+1)
Z = GF(q)
k = 1
while True:
    e0 = 0
    st = g^k % q
    if is_prime(st) and st not in baza:
        st = abs(st-q)
        e0 = 1
    tmp = ecm.factor(st)
    if set(tmp).issubset(set(baza)):
        relacija = [0]*(len(baza)+1)
        relacija[0] = e0
        for i in range(1, len(baza)):
            relacija[i] = tmp.count(baza[i])
        relacija[-1] = k

        seznam_relacij.append(relacija)
        M = Matrix(Z, seznam_relacij)
        if len(V.linear_dependence(M)) > 0:
            seznam_relacij = seznam_relacij[:-1]

            if len(seznam_relacij) > len(baza):
                break
    k+=1

tmp = seznam_relacij[:]
b = [0]*(len(baza)+1)
for i in range(len(baza)+1):
    if seznam_relacij[i][0] == 1:
        b[i] = seznam_relacij[i][-1] - ((q-1)//2)
    else:
        b[i] = seznam_relacij[i][-1]
tmp[i] = tmp[i][1:-1]

M = Matrix(IntegerModRing(q-1), tmp)
B = vector(IntegerModRing(q-1), b)
X = M.solve_right(B)
print(X)

j = 0
while True:
    rez = (power_mod(g, j, q)*h) % q

```

```

tmp = ecm.factor(rez)
if set(tmp).issubset(set(baza)):
    enacba = [0]*(len(baza)-1)
    for i in range(1,len(baza)):
        enacba[i-1] = tmp.count(baza[i])
    enacba = vector(IntegerModRing(q-1),enacba)

    return( enacba*X - j)
j+=1

```

## 8 MOV

MOV napad, poimenovan po njegovih avtorjih Menezes, Okamoto in Vanstone, s pomočjo Weilovega parjenja pretvori problem diskretnega logaritma iz  $E(\mathbb{F}_q)$  v problem diskretnega logaritma nad  $\mathbb{F}_{q^m}^\times$  [18]. Na ta način se izognemo računanju nad krivuljo. Nov problem diskretnega logaritma pa lahko sedaj rešimo z različnimi napadi, med drugim tudi z napadom Index-Calculus 7.1. MOV napad deluje če velikost polja  $\mathbb{F}_{q^m}$  ni dosti večja od velikosti polja  $\mathbb{F}_q$ . Postopek napada sledi poteku dokaza naslednje trditve.

### Trditev 8.1.

*Naj bo  $E$  eliptična krivulja nad  $\mathbb{F}_q$ . Naj bosta  $P, Q \in E(\mathbb{F}_q)$ , ter naj bo  $N$  red točke  $P$ . Predpostavimo, da velja  $\gcd(N, q) = 1$ . Potem obstaja tako število  $k$ , da velja  $Q = kP$  natanko tedaj ko  $NQ = \infty$  in  $e_N(P, Q) = 1$ .*

*Dokaz.*

( $\Rightarrow$ ) Če je  $Q = kP$ , potem je  $NQ = kNP$ , ampak ker je red  $P$  enak  $N$  od tod sledi  $kNP = \infty$ . Prav tako

$$e_n(P, Q) = e_n(P, P)^k = 1^k = 1.$$

( $\Leftarrow$ ) Naj bo  $NQ = \infty$ , torej je po definiciji  $Q \in E[N]$ . Ker je  $\gcd(N, q) = 1$  lahko uporabimo izrek 5.2 in zapišemo  $E[N] \cong \mathbb{Z}_N \oplus \mathbb{Z}_n$ . Sedaj izberemo točko  $R$  tako, da je  $\{P, R\}$  baza  $E[N]$ . Ker sta  $P, R$  baza lahko  $Q$  zapišemo kot

$$Q = aP + bR,$$

za neki števili  $a, b \in \mathbb{N}$ . Po posledici definicije Weilovega parjenja 6.3 velja  $e_N(P, R) = \zeta$  je generator  $\mu_N$ . Po predpostavki velja  $e_N(P, Q) = 1$  dobimo torej

$$1 = e_N(P, Q) = e_N(P, P)^a e_N(P, R)^b = \zeta^b.$$

Od tod sledi, da je  $b$  večkratnik števila  $N$ , od tod pa po definiciji sledi  $bR = \infty$ , ter  $Q = aP$ .  $\square$

Ideja dokaza nam sedaj da korake MOV napada.

---

**Algoritem 4** MOV napad

---

Vhod: Točki  $P$  in  $Q$  na eliptični krivulji  $E$ .

Izhod: diskretni logaritem  $k$ .

Izberi  $m$  tako, da

$$E[N] \subset E(\mathbb{F}_{q^m}).$$

Ker imajo vse točke  $E[N]$  koordinate v  $\overline{\mathbb{F}_q} = \cup_{j \geq 1} \mathbb{F}_{q^j}$  tak  $m$  obstaja. Prav tako je  $\mu_N$  v  $\mathbb{F}_{q^m}$ . Nato postopaj po naslednjih korakih.

1. Naključno izberi točko  $T \in E(\mathbb{F}_{q^m})$ .
  2. Izračunaj red  $M$  točke  $T$ .
  3. Naj bo  $d = \gcd(M, N)$  in naj bo  $T_1 = (M/d)T$ . Potem ima  $T_1$  red, ki deli  $N$ , torej je  $T_1 \in E[N]$ .
  4. Izračunaj  $\zeta_1 = e_N(P, T_1)$  in  $\zeta_2 = e_N(Q, T_1)$ . Tu sta  $\zeta_1$  in  $\zeta_2$  v  $\mu_d \subset \mathbb{F}_{q^m}^\times$ .
  5. Reši problem diskretnega logaritma  $\zeta_2 = \zeta_1^k$  v  $\mathbb{F}_{q^m}^\times$ . To nam da  $k \pmod d$ .
  6. Ponovi korake 1-5 za različne točke  $T$  dokler ni  $k$  določen.
- 

MOV napad deluje hitreje, kot če hočemo rešiti problem diskretnega algoritma direktno nad krivuljo, če velja

$$k > \log^2(p),$$

kjer krivuljo  $E(\mathbb{F}_p)$  MOV napad pretvori v grupo  $\mathbb{F}_{p^k}^\times$ .

MOV napad deluje na supersingularnih krivuljah, saj lahko za take krivulje ponavadi lahko vzamemo  $m = 2$ .

**Definicija 8.2.**

Eliptična krivulja  $E$  s karakteristiko  $p$  je *supersingularna* če je  $E[p] \cong 0$ .

**Izrek 8.3.**

Naj bo  $E$  eliptična krivulja nad  $\mathbb{F}_q$ , kjer je  $q$  potenca nekega praštevila  $p$ . Potem velja  $\#E(\mathbb{F}_q) = q + 1 - a$  za neko število  $a$ . Krivulja  $E$  je supersingularna natanko tedaj ko velja  $a \equiv 0 \pmod p$ , kar se zgodi natanko tedaj ko  $\#E(\mathbb{F}_q) \equiv 1 \pmod p$ .

*Dokaz.*

Zapišimo  $X^2 - aX + q = (X - \alpha)(X - \beta)$ . Trditvi 3.35 pravi, da velja

$$\#E(\mathbb{F}_{q^n}) = q^n + 1 - (\alpha^n + \beta^n).$$

Lema 3.36 pravi, da za  $s_n = \alpha^n + \beta^n$  velja rekurzivna zveza

$$s_0 = 2, \quad s_1 = a, \quad s_{n+1} = as_n - qs_{n-1}.$$

Predpostavimo sedaj, da velja  $a \equiv 0 \pmod p$ . Potem velja  $s_1 = a \equiv 0 \pmod p$  in  $s_{n+1} \equiv 0 \pmod p$ , za vse  $n \geq 1$ . Torej velja

$$\#E(\mathbb{F}_{q^n}) = q^n + 1 - s_n \equiv 1 \pmod p.$$

To pa ravno pomeni, da v  $E(\mathbb{F}_{q^n})$  ni točk reda  $p$  za noben  $n \geq 1$ . Ker lahko zapišemo  $\overline{\mathbb{F}_q}$  kot  $\overline{\mathbb{F}_q} \bigcup_{n \geq 1} \mathbb{F}_{q^n}$ , tudi v  $\overline{\mathbb{F}_q}$  nimamo nobene točke reda  $p$ . To pa po definiciji supersingularnosti pomeni, da je  $E$  supersingularna.

Dokažimo še drugo stran ekvivalence. Predpostavimo, da  $a \not\equiv 0 \pmod{p}$ . Rekurzivna zveza implicira  $s_{n+1} \equiv as_n \pmod{p}$ , za  $n \geq 1$ . Ker je  $s_1 = a$  imamo  $s_n \equiv a^n \pmod{p}$ , za vse  $n \geq 1$ . Od tod sledi

$$\#E(\mathbb{F}_{q^n}) = q^n + 1 - s_n \equiv 1 - a^n \pmod{p}.$$

Mali Fermatov izrek nam pove  $a^{p-1} \equiv 1 \pmod{p}$ . Zato ima  $E(\mathbb{F}_{q^{p-1}})$  red deljiv z  $p$ , kar pomeni, da vsebuje element reda  $p$ . To pa ravno pomeni, da  $E$  ni supersingularna.

Za zadnji del trditve pa sledi iz dejstvi

$$\#E(\mathbb{F}_q) = q + 1 - a \equiv 1 - a \pmod{p}.$$

Od tod sledi  $\#E(\mathbb{F}_q) \equiv 1 \pmod{p}$  natanko tedaj ko  $a \equiv 0 \pmod{p}$ .  $\square$

#### Opomba 8.4.

Izkaže se, da je pri pogoju  $q = p \geq 5$  ta definicija ekvivalentna temu, da je  $a = 0$ .

Naslednja trditev, nam bo dala utemeljitev izbire  $m$  pri supersingularnih krivuljah.

#### Trditev 8.5.

*Naj bo  $E$  eliptična krivulja nad  $\mathbb{F}_q$ , ter naj velja  $a = q + 1 - \#E(\mathbb{F}_q) = 0$ . Naj bo  $N \in \mathbb{N}$ . Če obstaja točka  $P \in E(\mathbb{F}_q)$  reda  $N$ , potem velja*

$$E[N] \subseteq E(\mathbb{F}_{q^2}).$$

*Dokaz.*

Frobeniusev endomorfizem  $\phi_q$  po izreku 3.34 zadošča  $\phi_q^2 - a\phi_q + q = 0$ . Ker velja  $a = 0$ , to pomeni

$$\phi_q^2 = -q.$$

Naj bo  $S \in E[N]$ . Ker velja  $\#E(\mathbb{F}_q) = q + 1$  in ker obstaja točka reda  $N$ , od tod sledi  $N | q + 1$ . Povedano drugače to pomeni  $-q \equiv 1 \pmod{N}$ . Od tod sledi

$$\phi_q^2(S) = -qS = 1 \cdot S.$$

Po lemi 3.26 od tod sledi  $S \in E(\mathbb{F}_{q^2})$ .  $\square$

#### Primer 8.6.

Denimo, da želimo rešiti problem diskretnega logaritma na krivulji  $E : y^2 = x^3 + x \pmod{547}$ , podanega s točkama  $P = (67, 481), Q = (167, 405)$ . Iščemo  $k$  tako da bo veljalo  $P = kQ$ .

Najprej moramo prvotno polje razširiti z nekim  $m$ . Ker je podana krivulja supersingularna, lahko za  $m$  izberemo 2. Krivuljo torej razširimo na polje  $\mathbb{F}_{547^2}$ . Vzemimo za nerazcepni polinom stopnje 2 polinom  $x^2 + 543x + 2$ . Elementi tega polja so torej polinomi stopnje 1 s koeficienti v  $\mathbb{F}_p$ . Sedaj moramo izbrati neko

naključno točko  $T$ . Denimo, da smo izbrali točko  $T = (24x + 219, 273x + 466)$ . Z algoritmom 5 sedaj poračunamo red  $M$  točke  $T$ . Kot odgovor dobimo  $M = 274$ . Poračunati moramo tudi red  $N$  točke  $P$ . Ta pa je enak  $N = 137$ . Največji skupni delitelj teh števil je 137. Zaradi tega dobimo novo točko  $T1$  kot

$$T1 = (M/d)T = (274/137)T = 2T = (440x + 318, 363x + 296).$$

Ko izračunamo to točko sedaj s pomočjo Weilovega parjenja poračunamo

$$e_{137}(P, T1) = 50x + 422, \quad e_{137}(Q, T1) = 416x + 519.$$

Tu moramo sedaj rešiti problem diskretnega logaritma v polju  $\mathbb{F}_{547^2}$ . To lahko naredimo s posplošitvijo index calculusa na končna polja [10]. Rešitev tega problema je 83. Rešitev lahko preverimo. Če poračunamo

$$(50x + 422)^{83} \div (x^2 + 543x + 2),$$

bi kot ostanek v  $\mathbb{F}_{547^2}$  res dobili polinom  $416x + 519$ . Postopek z naključno izbranimi  $T$ , bi sedaj še nekajkrat ponovili. Od tod pa bi potem lahko zaključili, da je iskani  $k = 83$ . Ta rezultat pa lahko s pomočjo algoritma za seštevanje točk tudi hitro preverimo.

## 8.1 Mali korak, Velik korak

V tem razdelku bomo opisali, kako izračunamo red točke na nek ekonomičen način. Trenutno je najhitrejši algoritem za izračun reda točke Schoof–Elkies–Atkinsonov algoritem (SEA) [14]. Tu pa si bomo pogledali malo bolj preprost algoritem, ki pa vseeno predstavlja veliko izboljšanje glede na naiven pristop seštevanja točke same s seboj. Naj bo  $P \in E(\mathbb{F}_q)$ . Radi bi izračunali red točke  $P$ . Iščemo torej tako število  $k$ , da bo veljalo  $kP = \infty$ . Algoritem Mali korak, Velik korak lahko izračuna red točke v približno  $4q^{\frac{1}{4}}$  korakih. Koraki algoritma so sledeči:



---

**Algoritem 5** Mali korak, Velik korak

---

Vhod: točka  $P$  na eliptični krivulji  $E$ .

Izhod: red točke  $P$ .

1. Izračunaj  $Q = (q + 1)P$ .
  2. Izberi število  $m$  za katero velja  $m > q^{\frac{1}{4}}$ . Za  $j = 0, 1, \dots, m$  izračunaj in shrani  $jP$ .
  3. Za  $k = -m, -m + 1, \dots, m - 1, m$  izračunaj točke  $Q + k(2mP)$ . V primeru, da se kakšna od teh točk ujema z  $\pm jP$  prekini računanje in si zapomni ustrezna  $k, j$ .
  4. Izračunaj  $(q + 1 + 2mk \mp j)P$  in poglej v katerem primeru je to enako  $\infty$ . V tem primeru naj bo  $M = (q + 1 + 2mk \mp j)$ .
  5. Faktoriziraj  $M$ . Označimo faktorje števila  $M$  z  $p_1, \dots, p_r$ .
  6. Izračunaj  $(M/p_i)P$ , za  $i = 1, \dots, r$ . Če je  $(M/p_i)P = \infty$  za nek  $i$ , potem zamenjaj  $M$  z  $M/p_i$  in pojdi nazaj na korak (5). V nasprotnem primeru je  $M$  red točke  $P$ .
- 

**Opomba 8.7.**

Algoritem 5 lahko preoblikujemo do te mere, da namesto reda točke izračunamo število točk na krivulji. Vse kar moramo narediti, je ponavljati korake (1)-(6) za naključno izbrane točke  $P \in E(\mathbb{F}_q)$ . Ustavimo se ko najmanjši skupni večkratnik redov točk, deli le eno število  $N$  v območju  $q + 1 - 2\sqrt{q} \leq N \leq q + q + 2\sqrt{q}$ . To število  $N$  je potem število točk na dani krivulji. Pri tem postopku se upremo na Hassejev izrek o številu točk na eliptični krivulji.

Tu se sedaj pojavi še vprašanje zakaj ta algoritem deluje. Odgovor na to vprašanje pa nam da naslednja lema.

**Lema 8.8.**

Naj bo  $G$  aditivna grupa in naj bo  $g \in G$ . Denimo da velja  $Mg = 0$  za nek  $M \in \mathbb{N}$ . Označimo s  $p_1, \dots, p_r$  različna praštevila, ki delijo  $M$ . Če velja  $(M/p_i)g \neq 0$  za vse  $i$ , potem je  $M$  red  $g$ .

*Dokaz.*

Naj bo  $k$  red  $g \in G$ . Ker velja  $Mg = 0$  vemo, da  $k|M$ . Denimo, da  $k \neq M$ . Pokazati moramo, da obstaja praštevilo  $p_i$ , tako da  $(M/p_i)g = 0$ . Naj bo  $p_i$  praštevilo, ki deli  $M/k$ . Tako število obstaja, ker  $M \neq k$ . Potem velja  $p_i k|M$ , to pa pomeni da  $k|(M/p_i)$ . Ker pa je  $k$  red elementa  $g$ , to pomeni  $(M/p_i)g = 0$ . □

**Primer 8.9.**

Vzemimo problem iz prejšnjega primera. Zanima nas torej red točke  $P = (67, 481)$ , ki se nahaja na krivulji  $y^2 = x^3 + x \pmod{547}$ .

Najprej poračunamo točko  $Q = (q + 1)P = 548P = \infty$ . Ker je  $Q$  točka v neskončnosti lahko izpustimo iskanje pravega  $k$  iz algoritma, saj bo pri  $k = 0, j = 0$  veljalo

$$Q + k(2mP) = Q = \infty = jP.$$

Za število  $M$  prav tako v obeh primerih dobimo isti rezultat in sicer  $M = 548$ . Velja  $548 = 2 \cdot 2 \cdot 137$ . Pri izvajanju nadaljnjih korakov algoritma število  $M$  dvakrat delimo z 2, saj velja  $274P = \infty$ ,  $137P = \infty$ . Ker pa je 137 praštevilo in  $P \neq \infty$  od tod sledi red  $N$  točke  $P$  je enak 137.

## 8.2 MOV napad program

Algoritem MOV sprogramiran v programskem okolju SAGE.

```
def MOV(E,q,m,P,Q,st_korakov = 10):
    """
    Opis:
        Funkcija MOV resuje problem diskretnega logaritma
        na krivulji E, podanega s točkama P,Q.
        Iscemo torej $k$, tako da velja $kP = Q$.

    Definicija:
        MOV(E,q,m,P,Q,st_korakov = 10)

    Vhodni podatki:
        E... Eliptična krivulja iz SAGE
        q... modul krivulje
        m... stevilo ki pove razširitev prvotnega polja
            iz GF(q) gremo v GF(q^m) (GF = Galvajevo polje)
        P... točka na eliptični krivulji iz SAGE
        Q... točka na eliptični krivulji iz SAGE
        st_korakov... v algoritmu dobivamo rezultate po nekem
            modulu in moramo na koncu s pomočjo
            kitajskega izreka dobiti končni rezultat.
            Lahko pa se zgodi da vedno dobivamo po
            istih moduli in se program ne bi končal.
            st_korakov omeji kolikokrat se ta del
            zanke izvede.

    Izhodni podatek:
        stevilo k
    """
    N = P.order()
    k = GF(q^m, 'x')
    E1 = E.base_extend(k) # razširimo krivuljo na večje polje
    P1 = E1(P) # razširimo moramo tudi točki
    Q1 = E1(Q)
```

```

odgovori = [[] , [[]]
koraki = 0

while True:
    #vsak ta del nam da odgovor po modolu d
    T = E1.random_element()
    M = T.order()
    d = gcd(N,M)
    T1 = int(M/d)*T
    w1 = P1.weil_pairing(T1,N)
    print(w1)
    w2 = Q1.weil_pairing(T1,N)
    print(w2)
    if w1 != w2:
        #k = Naivno(w1, w2)
        k = discrete_log(w2, w1)
        odgovori[0].append(k)
        odgovori[1].append(d)
    if lcm(odgovori[1]) >= q:
        break
    elif koraki > st_korakov:
        break
    koraki+=1
#uporabimo kitajski izrek o ostankih
k = crt(odgovori[0] , odgovori[1])
return k

```

## 9 Anomalne krivulje

Delovanje MOV temelji na uporabi Weilovega parjenja. Pojavi se torej ideja, da bi konstruirali tako krivuljo, ki vedno da trivialno Weilovo parjenje. S tem bi preprečili MOV napad. Krivulje za katere je to res so take, da za njih velja

$$\#E(\mathbb{F}_q) = q.$$

Takim krivuljam rečemo *anomalne krivulje*. V primeru, ko je  $p$  praštevilo, velja da je  $E[p]$  ciklična grupa, kar pomeni da bo Weilovo parjenje konstantno enako 1. Vendar za tak tip krivulj obstaja napad, ki deluje še hitreje kot MOV. Pokažimo algoritem, ki deluje če je  $q$  praštevilo. Za tak napad bomo morali najprej krivuljo  $E$  in točki  $P, Q$  razširiti iz  $\mathbb{F}_p$  na  $\mathbb{Z}$ . Pri tem bomo potrebovali naslednjo trditev.

### Trditev 9.1.

*Naj bo  $E$  eliptična krivulja nad  $\mathbb{F}_p$  in naj bosta  $P, Q \in E(\mathbb{F}_p)$ . Predpostavimo še, da je krivulja  $E$  oblike  $y^2 = x^3 + Ax + B$ . Potem obstajajo cela števila  $A', B', x_1, x_2, y_1, y_2$  in eliptična krivulja  $E'$  podana z*

$$y^2 = x^3 + A'x + B',$$

*ter točke  $P' = (x_1, y_1), Q' = (x_2, y_2) \in E'(\mathbb{Q})$ . Za ta števila velja*

$$A \equiv A', B \equiv B', P \equiv P', Q \equiv Q' \pmod{p}.$$

*Dokaz.*

Izberi števili  $x_1, x_2$  tako, da velja  $x_1, x_2 \pmod{p}$  podajata x-koordinati  $P, Q$ . Predpostavimo najprej, da  $x_1 \not\equiv x_2 \pmod{p}$ . Nato izberi  $y_1$  tako, da velja  $P' \equiv P \pmod{p}$ . Tu ekvivalenco gledamo po koordinatah. Pri izbiri  $y_2$  moramo biti bolj previdni. Izberimo  $y_2$  tako, da velja

$$y_2^2 \equiv y_1^2 \pmod{x_2 - x_1} \text{ in } (x_2, y_2) \equiv Q \pmod{p}.$$

To je mogoče, saj lahko uporabimo Kitajski izrek o ostankih, ker velja  $\gcd(p, x_2 - x_1) = 1$ .

Na ta način dobimo dve enačbi s pomočjo katerih moramo določiti konstanti  $A', B'$ . Enačbi se glasita

$$\begin{aligned} y_1^2 &= x_1^3 + A'x_1 + B, \\ y_2^2 &= x_2^3 + A'x_2 + B. \end{aligned}$$

Rešitev se torej glasi

$$A' = \frac{(y_2^2 - y_1^2) - (x_2^3 - x_1^3)}{x_2 - x_1}, B' = y_1^2 - x_1^3 - A'x_1.$$

Ker je  $y_2^2 - y_1^2$  po konstrukciji deljivo z  $x_2 - x_1$  in ker so vse ostale konstante cela števila, od tod sledi, da sta tudi  $A', B' \in \mathbb{Z}$ . Točki  $P', Q'$  pa ležita na  $E'$  po konstrukciji.

Obravnavati moramo še primer če je  $x_1 \equiv x_2 \pmod{p}$ . V tem primeru velja  $P = \pm Q$ . Izberimo potem  $x_1 = x_2$ , ter  $y_1$  tako, da nam  $y_1 \pmod{p}$  podaja y-koordinato točke  $P$ . Na isti način izberemo še  $A'$ , ter določimo  $B' = y_1^2 - x_1^3 - A'x_1$ .  $Q'$  pa izberemo kot  $\pm P'$ .

Preveriti moramo še, da enačba ki predstavlja  $E'$  res podaja eliptično krivuljo. Ker je krivulja  $E$  eliptična velja

$$4A^3 + 27B^2 \neq 0.$$

To pa pomeni

$$4A'^3 + 27B'^2 \equiv 4A^3 + 27B^2 \not\equiv 0 \pmod{p}.$$

Velja torej  $4A'^3 + 27B'^2 \neq 0$ . Kar pa pomeni, da je  $E'$  res eliptična krivulja.  $\square$

### Definicija 9.2.

Naj bosta  $a, b$  racionalni števili za kateri velja  $a/b \neq 0$  in  $a, b$  tuji števili. Zapišimo  $a/b = p^r a_1/b_1$ , kjer  $p \nmid a_1 b_1$ . Potem je  $p$ -adična vrednost definirana kot

$$v_p(a/b) = r.$$

Definirajmo še

$$v_p(0) = \infty.$$

### Primer 9.3.

$\frac{7}{40} = 2^{-3} \frac{7}{5}$ , zato velja

$$v_2(7/40) = -3.$$

Poglejmo si še kakšen primer

$$v_5(50/3) = 2, \quad v_7(1/2) = 0.$$

Naj bo  $E'$  eliptična krivulja nad  $\mathbb{Z}$  podana z  $y^2 = x^3 + A'x + B'$ . Naj bo  $r \in \mathbb{N}$ . Potem lahko definiramo

$$E'_r = \{(x, y) \in E'(\mathbb{Q}) \mid v_p(x) \leq -2r, v_p(y) \leq -3r\} \cup \{\infty\}.$$

### Izrek 9.4 ([18], Izrek 5.8).

Naj bo  $E'$  podana z  $y^2 = x^3 + A'x + B'$ ,  $A', B' \in \mathbb{Z}$ . Naj bo  $p$  praštevilo in naj bo  $r \in \mathbb{N}$ . Potem velja

1.  $E'_r$  je podgrupa  $E'(\mathbb{Q})$ .
2. Če je  $(x, y) \in E'(\mathbb{Q})$ , potem  $v_p(x) < 0$  natanko tedaj ko  $v_p(y) < 0$ . V tem primeru obstaja število  $r \geq 1$ , za katero velja  $v_p(x) = -2r, v_p(y) = -3r$ .
3. Preslikava

$$\begin{aligned} \lambda_r : E'_r/E'_{5r} &\rightarrow \mathbb{Z}_{p^{4r}} \\ (x, y) &\mapsto p^{-r}x/y \pmod{p^{4r}} \\ \infty &\mapsto 0 \end{aligned}$$

je injektivni homomorfizem.

4. Če  $(x, y) \in E'_r$  ampak  $(x, y) \notin E'_{r+1}$ , potem  $\lambda_r(x, y) \not\equiv 0 \pmod{p}$ .

Potrebovali bomo še preslikavo redukcije po modulu  $p$

$$\begin{aligned} \text{red}_p : E'(\mathbb{Q}) &\rightarrow E' \pmod{p} \\ (x, y) &\mapsto (x, y) \pmod{p} \\ E'_1 &\mapsto \{\infty\} \end{aligned}$$

Ta preslikava je homomorfizem, z jedrom  $E'_1$ .

Vrnimo se sedaj k prvotnemu problemu. Imamo torej anomalno krivuljo  $E$  nad  $\mathbb{F}_p$ . Želimo poiskati  $k$ , tako da bo veljalo  $Q = kP$ . Sledeči algoritem nam da rešitev tega problema.

---

**Algoritem 6** Teoretični algoritem nad anomalnimi krivuljami

---

Vhod: Točki  $P, Q$  nad eliptično krivuljo  $E$ .

Izho: diskretni logaritem  $k$ .

---

1. Razširi  $E, P, Q$  nad  $\mathbb{Z}$ , kot v trditvi 9.1.
  2. Naj bo  $P'_1 = pP', Q'_1 = pQ'$ .
  3. Če  $P'_1 \in E'_2$ , izberi nove  $E', P', Q'$  in se vrni na korak 2. V nasprotnem primeru je  $l_1 = \lambda_1(P'_1), l_2 = \lambda_1(Q'_1)$ . Iskani  $k$  pa je  $k \equiv l_1/l_2 \pmod{p}$ .
- 

### Opomba 9.5.

Drugi korak algoritma nam zagotavlja, da so  $P'_1, Q'_1 \in E'_1$ . To je res, ker smo na anomalni krivulji in velja  $\text{red}_p(pP') = p \cdot \text{red}_p(P') = \infty$ .

Tu je potrebno podati še utemeljitev, da tak napad res deluje. Označimo  $K' = kP' - Q'$ . Velja

$$\infty = kP - Q = \text{red}_p(kP' - Q') = \text{red}_p(K').$$

To ravno pomeni, da  $K' \in E'_1$ . Od tod pa sledi, da je  $\lambda_1(K')$  definiran in velja

$$\lambda_1(pK') = p\lambda_1(K') \equiv 0 \pmod{p}.$$

Torej velja

$$kl_1 - l_2 = \lambda_1(kP'_1 - Q'_1) = \lambda_1(kpP' - pQ') = \lambda_1(pK') \equiv 0 \pmod{p}.$$

To pa je ravno to kar hočemo pokazati. Izkaže se, da je ta algoritem nepraktičen, saj ima x-koordinata točk ponavadi okoli  $p^2$  števk. Vendar pa lahko problem rešimo, če delamo po modulu  $p^2$  namesto da delamo v  $\mathbb{Q}$ . Tako pridemo, do novega algoritma

---

**Algoritem 7** Napad na anomalne krivulje

---

Vhod: Točki  $P, Q$  nad eliptično krivuljo  $E$ .

Izho: diskretni logaritem  $k$ .

1. Razširi  $E, P, Q$  nad  $\mathbb{Z}$ , kot v trditvi 9.1.
2. Izračunaj  $P'_2 = (p-1)P' \equiv (x', y') \pmod{p^2}$ .
3. Izračunaj  $Q'_2 = (p-1)Q' \equiv (x'', y'') \pmod{p^2}$ .
4. Izračunaj

$$m_1 = p \frac{y' - y_1}{x' - x_1}, \quad m_2 = p \frac{y'' - y_2}{x'' - x_2}.$$

5. Če  $v_p(m_2) < 0$  ali  $v_p(m_1) < 0$  poizkusi na drugi krivulji  $E'$ . V nasprotnem primeru je  $k \equiv m_1/m_2 \pmod{p}$ .
- 

Utemeljitev delovanja algoritma lahko najdemo v [18].

**Primer 9.6.**

Naj bo krivulja  $E$  podana z enačbo  $y^2 = x^3 + 154x + 82$  nad poljem  $\mathbb{F}_{163}$ . Naj bosta  $P = (7, 6), Q = (150, 152)$  točki na krivulji. Iščemo tak  $k$ , da bo veljalo  $kP = Q$ . Preden lahko uporabimo algoritem 7 se moramo prepričati, da je  $E$  res anomalna krivulja. Za točko  $P$  velja  $163P = \infty$ . Ker je 163 praštevilo od tod sledi, da je red točke  $P$  enak 163. To pa pomeni, da  $163 | \#E(\mathbb{F}_{163})$ . Hassejev izrek 3.31 nam pove

$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q}, \\ 139 \leq \#E(\mathbb{F}_{163}) \leq 189.$$

Hkrati pa vemo, da red točke deli red grupe, kar pomeni, da je  $\#E(\mathbb{F}_{163}) = 163$ . To velja, ker je 163 edino število v tem območju, ki je deljivo z 163.

Sedaj lahko uporabimo zgornji napad. Razširimo krivuljo in točke nad  $\mathbb{Q}$  in dobimo

$$E' : y^2 = x^3 - 11908x + 83049, \quad P = (7, 6), \quad Q = (150, 1293).$$

Od tod potem poračunamo

$$P2 = 162 \cdot P1 = (12884, 24607) \pmod{26569}$$

$$Q2 = 162 \cdot Q1 = (10908, 3108) \pmod{26569}$$

Prav tako imamo

$$m_1 = 163 \frac{24607 - 6}{12884 - 7} = \frac{24601}{79} \\ m_2 = 163 \frac{3108 - 1293}{10908 - 150} = \frac{55}{2} \pmod{26569}$$

Velja torej

$$k = \frac{m_1}{m_2} = \frac{49202}{4345} \equiv 47 \pmod{163}.$$

## 9.1 Anonomalne program

```
from base import *
import random
from fractions import Fraction

def Lift(P,Q,meja = 20):
    """
    Opis:
        Funkcija Lift sprejme točki P,Q in ju
        vlozi v  $\mathbb{Z}$ , tako da so koordinate
        modulo p se vedno enake.

    Definicija:
        Lift(P,Q,meja = 20)

    Vhodni podatki:
        P... razred Point(EllipticCurve), ki
            predstavlja točko na elipticni
            krivulji in je definiran v dodatku base
        Q... razred Point(EllipticCurve), ki
            predstavlja točko na elipticni
            krivulji in je definiran v dodatku base
        meja... predsavlja zgornjo mejo iskanja
            naključnega stevila v algoritmu

    Izhodni podatek:
        Točki P,Q vloženi v  $\mathbb{Z}$ 
    """
    p = P.mod
    x1 = P.x + random.randint(0,meja)*p
    x2 = Q.x + random.randint(0,meja)*p
    if P.x != Q.x:
        y1 = P.y + random.randint(0,meja)*p
        razl = abs(x2-x1)
        i=0
        while True:
            y2 = Q.y + i*p
            if pow(y2,2,razl) == pow(y1,2,razl):
                break
            i += 1
        A1 = int((y2**2-y1**2)/(x2-x1)/
                -(x2**3-x1**3)/(x2-x1))
        B1 = y1**2-x1**3-A1*x1
    else:
```



```

        x2 = x1
        y1 = P.y + random.randint(0,meja)*p
        A1 = A + random.randint(0,meja)*p
        B1 = y1**2-x1**3-A1*x1
        if P.y == Q.y:
            y2 = y1
        else:
            y2 = -y1
    return (Point(A1,B1,p**2,x1,y1),Point(A1,B1,p**2,x2,y2))

def AnomalneAttack(P,Q):
    """
    Opis:
        Funkcija AnomalneAttack izvede resuje
        problem diskretnega logaritma
        $$kP = Q$$
        nad anomalnimi krivuljami

    Definicija:
        AnomalneAttack(P,Q)

    Vhodni podatki:
        P... razred Point(EllipticCurve), ki
        predstavlja točko na elipticni
        krivulji in je definiran v dodatku base
        Q... razred Point(EllipticCurve), ki
        predstavlja točko na elipticni
        krivulji in je definiran v dodatku base

    Izhodni podatek:
        diskretni logaritem točke $P$
    """
    (P1,Q1) = Lift(P,Q)
    p = P.mod
    P2 = (p-1)*P1
    Q2 = (p-1)*Q1
    m1 = p*Fraction(P2.y-P1.y,P2.x-P1.x)
    m2 = p*Fraction(Q2.y-Q1.y,Q2.x-Q1.x)
    k = m1/m2
    k1 = k.numerator
    k2 = k.denominator
    k2 = NumberMod(k2,p).inverse().num
    k = (k1*k2) % p
    return k

```

## 10 Kriptografija nad Eliptičnimi krivuljami

### 10.1 ElGamalov kriptosistem

Opišimo najprej na kratko pojema javnega in privatnega ključa. V asimetričnih kriptosistemih kjer imamo za šifriranje in dešifriranje različna ključa, ključ za šifriranje javno objavimo. Na ta način lahko vsak z našim javnim ključem pošlje šifrirano sporočilo, dešifriramo pa ga lahko le z našim privatnim (dešifrirnim) ključem.

ElGamalova enkripcija javnega ključa temelji na problemu diskretnega logaritma. V primeru, da bi rešili problem diskretnega logaritma, bi lahko prebrali tudi vsa poslana sporočila. Oglejmo si kako izgleda ElGamalov kriptosistem. Denimo, da hoče Alenka komunicirati z Borisom, pri čemer ima Boris javno objavljeno poljubno izbrano krivuljo  $E$  nad nekim končnim poljem  $\mathbb{F}_q$ , točko  $P$  na krivulji in točko  $B$  na krivulji. Borisov privatni ključ je v tem primeru nek naključno izbran  $s$  s katerim je poračunal točko  $B$ , kot  $B = sP$ . Tu je  $P$  naključno izbrana točka. Če hoče Alenka komunicirati z Borisom ta postopek poteka na sledeč način:

1. Predstavi sporočilo, kot neko točko  $M \in E(\mathbb{F}_q)$ .
2. Naključno izberi skrivno število  $k$  in izračunaj  $M_1 = kP$ .
3. Izračunaj  $M_2 = M + kB$ .
4. Pošlji Borisu  $M_1, M_2$ .

Boris izračuna  $M$  kot

$$M = M_2 - sM_1.$$

Prepričajmo se, da na ta način res dobimo točko  $M$ .

$$M_2 - sM_1 = (M + kB) - k(sP) = M + ksBP - ksP = M.$$

Vsak tretji poslušalec, ki bi znal rešiti problem diskretnega logaritma, bi lahko s pomočjo  $P, B$  izračunal  $s$ , ali pa s pomočjo  $P, M_1$  izračunal  $k$ .

Pomembno pri celotni shemi pa je tudi to, da Alenka za različna sporočila ne uporablja enakega števila  $k$ . Recimo, da bi za sporočili  $M, M'$  Alenka uporabila isti  $k$ . Vsak tretji poslušalec bi lahko opazil, da sta v tem primeru točki  $M_1, M'_1$  enaki. Tako bi lahko izračunal

$$M_2 - M'_2 = M' - M.$$

To na prvi pogled ne deluje resna grožnja, a bi lahko povzročila velika škodo, če delamo z informacijami ki bi kasneje postale javne. Če bi npr. sporočilo, ki je predstavljeno s točko  $M$  kasneje postalo javno bi lahko brez težav sedaj izračunali še

$$M' = M - M_2 + M'_2.$$

Vprašanje, ki ga moramo tu še razrešiti je kako sporočilo predstaviti kot točko na krivulji. Ena izmed možnosti je način, ki ga je predlagal Koblitz [11]. Denimo, da je krivulja podana z enačbo  $y^2 = x^3 + Ax + B$  nad  $\mathbb{F}_p$ . Sporočilo  $m$  predstavimo kot število  $0 \leq m \leq p/100$ . Sedaj določimo  $x_j = 100m + j$ , za  $0 \leq j < 100$ . Podobno, kot pri generiranju naključne točke sedaj poizkusimo določiti koordinato  $y_j$  če ta

obstaja. Če smo uspeli poračunati obe koordinati je naša točka  $M = (x_j, y_j)$ . Ker velja  $0 \leq x_j < p + 100$  je  $m$  določen kot  $m = \lfloor x_j \rfloor$ . Ocenimo lahko, da je verjetnost neobstoja take točke približno  $2^{-100}$ .

## 10.2 Kriptosistemi nad parjenji

V enem od prejšnjih poglavij smo videli, kako lahko s pomočjo Weilovega parjenja izvedemo napad na problem diskretnega logaritma nad eliptičnimi krivuljami. Supersingularne krivulje so bile med tistimi, na katerih ta napad deluje. Sedaj pa si bomo ogledali kriptosistem, ki uporablja supersingularne krivulje [6]. Tu se opremo na dejstvo, da kljub napadom kot je MOV, problem diskretnega logaritma v  $\mathbb{F}_p^\times$  ni enostaven. Če izberemo dovolj velik  $p$  je ta problem še vedno zelo zahteven. Take krivulje izbiramo zato, ker lahko izkoristimo določene lastnosti.

Delajmo sedaj nad krivuljo  $E$  podano z enačbo  $y^2 = x^3 + 1$  nad  $\mathbb{F}_p$ , kjer je  $p \equiv 2 \pmod{3}$ . Take krivulje so supersingularne. Naj bo  $\omega \in \mathbb{F}_{p^2}$  primitivni tretji koren enote. Definirajmo preslikavo

$$\beta : E(\mathbb{F}_{p^2}) \rightarrow E(\mathbb{F}_{p^2}), (x, y) \mapsto (\omega x, y), \beta(\infty) = \infty.$$

Predpostavimo, da ima  $P$  red  $n$ . Potem ima tudi  $\beta(P)$  red  $n$ . Uporabimo modificirano Weilovo parjenje

$$e'_n(P_1, P_2) = e_n(P_1, \beta(P_2)).$$

Pokazali smo že v lemi 7.5, da v primeru ko velja  $3 \nmid n$  in ima  $P \in E(\mathbb{F}_p)$  red  $n$ , potem je  $e'_n(P, P)$  primitivni  $n$ -ti koren enote.

Ker je  $E$  supersingularna ima red  $p + 1$ . Predpostavimo še, da velja  $p = 6l - 1$  za neko praštevilo  $l$ . To pomeni, da ima točka  $6P$  red  $l$  ali 1 za vsako točko  $P$ . Za pripravo kriptosistema najprej neka agencija, ki jamči našo identiteto naredi sledeče:

- Izbere veliko praštevilo  $p = 6l - 1$ .
- Izbere točko  $P$  reda  $l$  na  $E(\mathbb{F}_p)$ .
- Izbere zgoščevalni funkciji  $H_1, H_2$ . Funkcija  $H_1$  vzame niz bitov poljubne dolžine in vrne točko reda  $l$  na krivulji  $E$ . Funkcija  $H_2$  pa vzame element reda  $l$  iz  $\mathbb{F}_{p^2}^\times$  in vrne binarni niz dolžine  $n$ , kjer je  $n$  dolžina sporočila, ki bo poslano.
- Izbere skrivno število  $s \in \mathbb{F}_l^\times$  in izračuna  $P_{\text{javni}} = sP$ .
- Javno objavi  $p, H_1, H_2, n, P, P_{\text{javni}}$  obdrži pa  $s$ .

Če hoče sedaj uporabnik z identiteto  $I$  dobiti privatni ključ, agencija naredi sledeče:

- Izračuna  $Q_I = H_1(I)$ .
- Izračuna  $D_I = sQ_I$ .
- Ko preveri identiteto uporabnika  $I$  pošlje  $D_I$  uporabniku.

Če hoče sedaj Alenka Borisu poslati sporočilo  $M$  to naredi tako:

- Alenka poišče Borisovo identiteto  $I$ , ter izračuna  $Q_I = H_1(I)$ .
- Izbere naključni element  $r \in \mathbb{F}_l^\times$ .
- Izračuna  $g_I = e'_l(Q_I, P_{\text{javni}})$ .
- Sporočilo  $c$  zapiše kot

$$c = (rP, M \oplus H_2(g_I^r)).$$

Tu oznaka  $\oplus$  predstavlja operacijo XOR na bitih.

Boris sedaj sporočilo  $c = (u, v)$  dešifrira na sledeči način.

- S pomočjo privatnega ključa  $D_I$  izračunaj  $h_I = e'_l(D_I, u)$ .
- Izračunaj  $m = v \oplus H_2(h_I)$ .

Prepričajmo se, da na ta način res dobimo originalno sporočilo.

$$e'_l(D_I, u) = e'_l(sQ_I, rP) = e'_l(Q_I, P)^{sr} = e'_l(Q_I, P_{\text{javni}})^r = g_I^r.$$

Sporočilo pa je sedaj

$$m = v \oplus H_2(e'_l(D_I, u)) = (M \oplus H_2(g_I^r)) \oplus H_2(g_I^r) = M.$$

## 11 Zaključek

Skozi delo smo spoznali, da izredno pomembno vlogo pri varnosti kriptosistemov nad eliptičnimi krivuljami igra sama izbira krivulje. Ponavadi krivulje generiramo naključno, nato pa preverimo vse potrebne pogoje, kot so supersingularnost, red grupe, ... Tekom dela smo ugotovili, da supersingularne krivulje ne predstavljajo dobre izbire če hočemo delati z zelo majhnimi ključi, saj le te niso odporne na napade kot je MOV. Prav tako smo videli, da lahko problem diskretnega logaritma nad anomalnimi krivuljami rešimo zelo enostavno. Ker smo si ogledali samo majhen del krivulj in načinov kako lahko izkoristimo njihovo obliko zaključimo delo s primerom krivulje, ki trenutno velja za varno. Krivulja imenovana M-221 je podana z enačbo

$$E: y^2 = x^3 + 117050x^2 + x \pmod{2^{221} - 3}.$$

Predstavili pa so jo Aranha, Barreto, Pereira in Ricardini leta 2013 [5].

## A Program osnovne strukture kubičnih krivulj

```
import math
import random

def razEvk(a, b, inverz = True):
    """
    Opis:
        funkcija razEvk predstavlja razširjen
        Evklidov algoritem

    Definicija:
        razEvk(a, b, inverz = True)

    Vhodni podatki:
        a...prvo stevilo
        b...drugo stevilo
        inverz...ce zelimo racunati samo inverz
            ne potrebujemo dodatne vrednosti
            ki nam jo da razširjen Evklidov
            algoritem

    Izhodni podatek:
        stevili $gcd(a,b), x, y$ tako, da
        $ax+by = gcd(a,b)$
        ali v primeru ko je inverz True
        stevili $gcd(a,b), a^{-1}$

    """
    if inverz == False:
```

```

s0, s1, t0, t1 = 1, 0, 0, 1
while b != 0:
    q, a, b = a // b, b, a % b
    s0, s1 = s1, s0 - q * s1
    t0, t1 = t1, t0 - q * t1
return a, s0, t0
else:
    s0, s1 = 1, 0
    while b != 0:
        q, a, b = a // b, b, a % b
        s0, s1 = s1, s0 - q * s1
    return a, s0

class NumberMod:
    """
    Opis:
        Razred za predstavitev števil po nekem
        modulu p

    Definicija:
        NumberMod(a, mod)

    Vhodni podatki:
        a... stevilo, ki ga hocemo predstaviti
        mod... modul po katerem delamo

    Izhodni podatek:
        Razred števila po modulu
    """
    def __init__(self, a, n=None):
        self.original = a
        self.modul = n
        if self.modul == None:
            self.num = a
        else:
            self.num = (a % self.modul)
            self.numS = self.num
            self.small(self.modul)

    def __neg__(self):
        return NumberMod(-self.num, self.modul)

    def __add__(self, other):
        """seštevanje števil po modulu"""
        if self.modul != other.modul:
            raise Exception('numbers_have_different_modul')

```

```

    else:
        novonum = NumberMod((self.num + other.num)\
                               % self.modul, self.modul)
        return novonum

def __sub__(self, other):
    """odstevanje števil po modolu"""
    if self.modul != other.modul:
        raise Exception('numbers_have_different_modul')
    elif self.modul == None:
        return NumberMod(self.num - other.num, self.modul)
    else:
        novonum = NumberMod((self.num - other.num)\
                               % self.modul, self.modul)
        return novonum

def __mul__(self, other):
    """množenje števil po modolu"""
    if self.modul != other.modul:
        raise Exception('numbers_have_different_modul')
    else:
        novonum = NumberMod((self.num * other.num)\
                               % self.modul, self.modul)
        return novonum

def __truediv__(self, other):
    """množenje z inverzom števila po modolu"""
    if self.modul != other.modul:
        raise Exception('numbers_have_different_modul')
    else:
        temp = other.inverse()
        novonum = self*temp
        return novonum

def __lt__(self, other):
    """manjše kot"""
    if self.num > 0 and other.num > 0:
        return self.num < other.num
    elif self.num < 0 and other.num > 0:
        return (self.modul + self.num) < other.num
    elif self.num > 0 and other.num < 0:
        return self.num < (other.modul + other.num)
    else:
        return self.num < other.num

def __le__(self, other):

```

```

    """manjse ali enako"""
    if self.num > 0 and other.num > 0:
        return self.num <= other.num
    elif self.num < 0 and other.num > 0:
        return (self.modul + self.num) <= other.num
    elif self.num > 0 and other.num < 0:
        return self.num <= (other.modul + other.num)
    else:
        return self.num <= other.num

def __eq__(self, other):
    """enakost"""
    return self.num == other.num

def __ne__(self, other):
    """ne enakost"""
    return self.num != other.num

def __gt__(self, other):
    """vecje kot"""
    if self.num > 0 and other.num > 0:
        return self.num > other.num
    elif self.num < 0 and other.num > 0:
        return (self.modul + self.num) > other.num
    elif self.num > 0 and other.num < 0:
        return self.num > (other.modul + other.num)
    else:
        return self.num > other.num

def __ge__(self, other):
    """vecje ali enako"""
    if self.num > 0 and other.num > 0:
        return self.num >= other.num
    elif self.num < 0 and other.num > 0:
        return (self.modul + self.num) >= other.num
    elif self.num > 0 and other.num < 0:
        return self.num >= (other.modul + other.num)
    else:
        return self.num >= other.num

def __pow__(self, b):
    p = self.modul
    if not isinstance(b, int):
        raise Exception('power_must_be_an_integer')
    else:
        if b == 0:

```



```

        return NumberMod(1,p)
    elif b == 1:
        return NumberMod(self.num,p)
    else:
        if b%2 == 0:
            st = NumberMod(self.num*self.num,p)
            return st**(b//2)
        else:
            st = NumberMod(self.num,p)**(b-1)
            return NumberMod(self.num,p)*st

def __str__(self):
    return str(self.num)

def __repr__(self):
    return str(self.num)

def inverse(self):
    """inverz stevila po modolu"""
    if self.modul == None:
        raise Exception('modul_equals_None')
    n = self.modul
    if self.num < 0:
        a = self.num + n
    else:
        a = self.num
    if gcd(a,n) != 1:
        raise Exception('inverse_does_not_exist')
    else:
        return NumberMod(razEvk(a,n)[1],n)

def small(self,n):
    """ce je stevilo veliko ga proba funkcija zapisati
    s predznakom minus"""
    if n == None:
        pass
    else:
        if ((n-1)/2) < self.numS:
            self.numS = self.numS-n
        elif self.numS < 0 and -((n-1)/2) > self.numS:
            self.numS = self.numS+n

def isPrime(self,eps = 1/100000):
    """Miller-Rabinov test za prastevilstvo
    z napako eps"""

```

```

#Napaka posameznega koraka je manjša kot 1/4
k = int(math.log(eps)/math.log(1/4))+1
r = 0
n = self.num
if n == 3:
    return True
if n < 0:
    n += self.modul
if n % 2 == 0:
    return False
temp = n-1
d = temp
while temp % 2 == 0:
    r += 1
    d = int(d/2)
    temp = d
i = 0
while i < k:
    a = random.randint(2,n-2)
    xprej = pow(a,d,n)
    j = 1
    while j < r+1:
        xnov = pow(xprej,2,n)
        if (xnov % n == 1 and
            (xprej != n-1 and xprej != 1)):
            #xprej je Millerjeva prica
            return False
        j += 1
        xprej = xnov
    if xnov % n != 1:
        #xnov Fermatova prica
        return False
    i += 1
return True

def gcd(self,other):
    p = self.modul
    q = other.modul
    if p == q:
        return NumberMod(gcd(self.num,other.num),p)
    else:
        raise Exception('numbers_have_different_modul')

def gcd(a,b):
    """

```

*Opis:*

*Funkcija gcd poisce največji  
skupni delitelj števil \$a, b\$*

*Definicija:*

*gcd(a, b)*

*Vhodni podatki:*

*a...prvo stevilo*

*b...drugo stevilo*

*Izhodni podatek:*

*največji skupni delitelj števil  
\$a, b\$*

*"""*

**if** b == 0:

**return** a

**else:**

**return** gcd(b, a % b)

**class** EllipticCurve:

*"""*

*Opis:*

*Razred elipticnih krivulj v Weierstrassovi obliki  
modulo p  
 $y^2 = x^3 + ax + b \pmod p$*

*Definicija:*

*EllipticCurve(a, b, mod)*

*Vhodni podatki:*

*a...parameter a v enacbi*

*b...parameter b v enacbi*

*mod...modul p po katerem delamo*

*Izhodni podatek:*

*Razred elipticne krivulje*

*"""*

**def** \_\_init\_\_(self, a, b, modulo):

self.a = a

self.b = b

self.mod = modulo

elipticna = self.isElliptic()

**if not** elipticna:

```

        raise Exception( 'Curve_is_not_Eliptic' )
def __str__( self ):
    """Pri izpisu print nam vrne
    krivuljo v lepi obliki"""
    if self.a > 0 and self.b > 0:
        return "y^2=_x^3+_ {0}x+_ {1}_mod_ {2}" .format(
            self.a, self.b, self.mod)
    elif self.a > 0 and self.b <0:
        return "y^2=_x^3+_ {0}x_-_ {1}_mod_ {2}" .format(
            self.a, abs( self.b ), self.mod)
    elif self.a > 0 and self.b == 0:
        return "y^2=_x^3+_ {0}x_mod_ {1}" .format(
            self.a, self.mod)
    elif self.a == 0 and self.b > 0:
        return "y^2=_x^3+_ {0}_mod_ {1}" .format(
            self.b, self.mod)
    elif self.a == 0 and self.b == 0:
        return "y^2=_x^3_mod_ {0}" .format(
            self.mod)
    elif self.a == 0 and self.b <0:
        return "y^2=_x^3_-_ {0}_mod_ {1}" .format(
            abs( self.b ), self.mod)
    elif self.a < 0 and self.b >0:
        return "y^2=_x^3_-_ {0}x+_ {1}_mod_ {2}" .format(
            abs( self.a ), self.b, self.mod)
    elif self.a < 0 and self.b == 0:
        return "y^2=_x^3_-_ {0}x_mod_ {1}" .format(
            abs( self.a ), self.mod)
    elif self.a < 0 and self.b <0:
        return "y^2=_x^3_-_ {0}x_-_ {1}_mod_ {2}" .format(
            abs( self.a ), abs( self.b ), self.mod)

```

```

def rand( self ):
    """

```

*Opis:*

*Funkcija rand generira naključno točko na  
elipticni krivulji E  
 $y^2 = x^3 + ax + b \mod p$*

*Definicija:*

*E.rand()*

*Vhodni podatki:*

*ni vhodnih podatkov*

```

        Izhodni podatek:
        Razred Point, ki predstavlja točko na
        elipticni krivulji
        """
    najdl = False
    while not najdl:
        x = random.randint(0, self.mod-1)
        y2 = NumberMod(x, self.mod)**3\
            + NumberMod(self.a*x, self.mod)\
            + NumberMod(self.b, self.mod)
        y2 = y2.num
        if pow(y2, (self.mod-1)//2, self.mod) == 1:
            y = TonelliShanks(y2, self.mod)
            najdl = True
    return Point(self.a, self.b, self.mod, x, y)

def isOn(self, P):
    """
    Opis:
        Funkcija isOn preveri ali točka P res
        leži na elipticni krivulji E

    Definicija:
        E.isOn(P)

    Vhodni podatki:
        P... razred Point, ki predstavlja točko
        na elipticni krivulji

    Izhodni podatek:
        True/False
    """

    if (P.a != self.a or P.b != self.b
        or P.mod != self.mod):
        return False
    else:
        x = NumberMod(P.x, self.mod)**3\
            + NumberMod(self.a*P.x, self.mod)\
            + NumberMod(self.b, self.mod)
        y = (P.y**2) % self.mod
        return x.num==y
def isEliptic(self):
    """Preverimo ali je dana krivulja res elipticna"""
    if not (self.a == None and self.b == None and self.mod == None):

```

```

        #ce to velja je točka v neskončnosti nas ne zanima
        temp = (4*self.a**3 + 27*self.b**2) % self.mod
        if temp != 0:
            return True
        else:
            return False
    else:
        return True

INF = "inf"

class Point(EllipticCurve):
    """
    Opis:
        Razred točk na elipticni krivulji, ki je
        podrazred razreda EllipticCurve.

        Točka  $\infty$  je podana kot
        INFPoint = Point(None, None, None, INF, INF),
        kjer je spremenljivka INF = "inf"

    Definicija:
        Point(a, b, mod, x, y)

    Vhodni podatki:
        a...parameter a v enacbi
        b...parameter b v enacbi
        mod...modul p po katerem delamo
        x...x-koordinata točke
        y...y-koordinata točke

    Izhodni podatek:
        Razred točke na elipticni krivulji
    """

    def __init__(self, a, b, mod, x, y):
        super().__init__(a, b, mod)
        self.x = x
        self.y = y
        if ((self.x == INF or self.y == INF)
            and self.y != self.x):
            #preverimo ali je točka v neskončnosti,
            # v tem primeru zahtevamo, da sta
            # obe koordianti INF
            raise Exception('Point_not_infinity')

```

```

def __str__(self):
    """Za lepsi izpis tocke po klicu print"""
    if self.x != INF:
        return "({0},{1})_mod_{2}".format(
            self.x, self.y, self.mod)
    else:
        return u"\u221e"

def __repr__(self):
    """za lepsi izpis tocke po klicu return"""
    if self.x != INF:
        return "({0},{1})_mod_{2}".format(
            self.x, self.y, self.mod)
    else:
        return u"\u221e"

def __add__(self, Q):
    """Algoritem za sestevanje tock nad isto elipticno
    krivuljo."""
    infly = False
    if self.x == INF or Q.x == INF:
        infly = True

    if (not(self.a == Q.a and self.b == Q.b
            and self.mod == Q.mod)
        and not infly):
        raise Exception("""Points don't lie on the
            same curve, or have different modulus""")

    if self.x == INF:
        return Q
    elif Q.x == INF:
        return self

    elif self.x != Q.x:
        m = NumberMod(Q.x-self.x, self.mod).inverse().num
        #print("stevec: ", Q.y-self.y)
        m = (m*(Q.y-self.y)) % self.mod
        x3 = (m**2-self.x-Q.x) % self.mod
        y3 = (m*(self.x-x3) - self.y) % self.mod
        return Point(self.a, self.b, self.mod, x3, y3)

    elif self.x == Q.x and self.y != Q.y:
        return Point(None, None, None, INF, INF)

```

```

    elif (self.x == Q.x and self.y == Q.y
          and self.y != 0):
        m = NumberMod(2*self.y, self.mod).inverse().num
        m = (m*(3*(self.x**2)+self.a)) % self.mod
        x3 = (m**2-2*self.x) % self.mod
        y3 = (m*(self.x-x3) - self.y) % self.mod
        return Point(self.a, self.b, self.mod, x3, y3)

    else:
        return Point(None, None, None, INF, INF)

def __neg__(self):
    if self == INFPoint:
        return INFPoint
    else:
        return Point(self.a, self.b, self.mod,
                      self.x, -self.y % self.mod)

def __sub__(self, Q):
    return self + (-Q)

def __rmul__(self, num):
    """Funkcija predstavlja mnozenje stevila z tocko
    npr.  $5P = P+P+P+P+P$ """
    if not isinstance(num, int):
        raise Exception('Can only multiply with an int')

    if num < 0:
        return -(abs(num)*self)

    elif num == 0:
        return Point(None, None, None, INF, INF)
    elif num == 1:
        return self
    else:
        if num % 2 == 0:
            pol = int(num / 2)
            return (pol*self + pol*self)
        else:
            return (self+(num-1)*self)

def __eq__(self, Q):
    """Dve tocki sta enaki, ce lezita na

```



```

        isti krivulji in imata iste koordiante
        ali pa predstavljata točko v neskončnosti"""
    if self.x == INF and Q.x == INF:
        return True
    elif (self.a == Q.a and self.b == Q.b
          and self.x == Q.x and self.y == Q.y
          and self.mod == Q.mod):
        return True
    else:
        return False

```

```

INFPoint = Point(None, None, None, INF, INF)

```

```

def TonelliShanks(n, p1):
    """
    Opis:
        Funkcija TonelliShanks izracuna kvadraticni
        ostanek stevila $n$ modulo $p1$

    Definicija:
        TonelliShanks($n, p1$)

    Vhodni podatki:
        $n...$ stevilo katerega kvadraticni
        ostanek nas zanima
        $p1...$ modul po katerem delamo

    Izhodni podatek:
        kvadraticni ostanek stevila $n$,
        ce ta obstaja
    """
    p = p1-1
    s = 0
    while True:
        if p % 2 == 0:
            s += 1
            p = p // 2
        else:
            break
    Q = p

    while True:
        z = random.randint(0, p1-1)
        if pow(z, (p1-1)//2, p1) == p1-1:
            break
    M = s

```

```

c = pow(z,Q,p1)
t = pow(n,Q,p1)
R = pow(n,(Q+1)//2,p1)

while True:
    if t == 0:
        return 0
    elif t == 1:
        return R
    else:
        i = 1
        while i < M:
            if pow(t,pow(2,i),p1) == 1:
                break
            i += 1
        b = pow(c,pow(2,M-i-1),p1)
        M = i
        c = pow(b,2,p1)
        t = (t*pow(b,2,p1)) % p1
        R = (R*b) % p1

def lema(n,a,q):
    s0 = 2
    s1 = a
    if n == 1:
        return a
    else:
        i=1
        while i < n:
            s2 = a*s1-q*s0
            s0 = s1
            s1 = s2
            i +=1
        return s2

def NumberPoints(steviloOsnovne,potenca,q):
    a = q+1-steviloOsnovne
    sn = lema(potenca,a,q)
    stevilo = pow(q,potenca)+1-sn
    return stevilo

```

## Literatura

- [1] *Recommendation for Cryptographic Key Generation*.
- [2] *Euler's Criterion*, [https://proofwiki.org/wiki/Euler%27s\\_Criterion](https://proofwiki.org/wiki/Euler%27s_Criterion), 2020.
- [3] *Extended Euclidean algorithm*, [https://en.wikipedia.org/wiki/Extended\\_Euclidean\\_algorithm](https://en.wikipedia.org/wiki/Extended_Euclidean_algorithm), 2020.
- [4] *Zapiski predavanj predmeta Algebra II*, šolsko leto 2015/2016.
- [5] D. Aranha in dr., *A note on high-security general-purpose elliptic curves*, <https://eprint.iacr.org> (2013).
- [6] D. Boneh in M. Franklin, *Identity-Based Encryption from the Weil Pairing*, SIAM Journal on Computing (2003).
- [7] A. Clark, *Elements of Abstract Algebra*, Dover Publications, 1984.
- [8] C. Easttom, *Modern Cryptography: Applied Mathematics for Encryption and Information Security*, McGraw-Hill Education, 2015.
- [9] C. G. Gibson, *Elementary Geometry of Algebraic Curves*, Cambridge University press, 1999.
- [10] C. Henri in dr., *Handbook of Elliptic and Hyperelliptic Curve Cryptography (Discrete Mathematics and Its Applications)*, Chapman and Hall/CRC, 2005.
- [11] N. Koblitz, *Elliptic Curve Cryptosystem*, mathematics of computation volume 48 (1987).
- [12] H. D. Lenstra, *Factoring Integers with Elliptic Curves*, The Annals of Mathematics (1987).
- [13] V. S. Miller, *Short Programs for functions on Curves*, 1986.
- [14] R. Schoof, *Counting points on elliptic curves over finite fields*, Journal de Théorie des Nombres de Bordeaux (1995).
- [15] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Springer, druga izd., 2009.
- [16] D. R. Stinson, *Cryptography: Theory and Practice (Discrete Mathematics and Its Applications)*, CRC-Press, 1995.
- [17] G. Tornarí, *Square Roots Modulo  $p$* , v: LATIN 2002: Theoretical Informatics.
- [18] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*, Chapman and Hall/CRC, druga izd., 2008.

