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, 2019

Zahvala

Neobvezno. Zahvaljujem se ...

Kazalo

Program dela

Mentor naj napiše program dela skupaj z osnovno literaturo. Na literaturo se lahko sklicuje kot [?], [?], [?], [?].

Osnovna literatura

Literatura mora biti tukaj posebej samostojno navedena (po pomembnosti) in ne le citirana. V tem razdelku literature ne oštevilčimo po svoje, ampak uporabljamo okolje itemize in ukaz plancite, saj je celotna literatura oštevilčena na koncu.

[?]

[?]

[?]

[?]

Podpis mentorja:

Kubične krivulje v kriptografiji

Povzetek

Tukaj napišemo povzetek vsebine. Sem sodi razlaga vsebine in ne opis tega, kako je delo organizirano.

English translation of the title

Abstract

An abstract of the work is written here. This includes a short description of the content and not the structure of your work.

Math. Subj. Class. (2010): oznake kot 74B05, 65N99, na voljo so na naslovu `http://www.ams.org/msc/msc2010.html?t=65Mxx`

Ključne besede: kubična krivulja , kriptografija , ...

Keywords: cubic curve , cryptography

1 Uvod

Kubične krivulje se v kriptografiji uporabljajo, ker zagotavljajo isto varnost, kot drugi klasični kriptosistemi, pri tem pa potrebujejo manjšo velikost ključa. Ocenjuje se, da je 2048 bitni ključ v RSA algoritmu enako varen kot 224 bitni ključ nad kubičnimi krivuljami. Krajši kluči predstavljajo veliko prednost v okoljih s slabšo procesorsko močjo in/ali omejenim pomnilnikom. Primer take uporabe predstavljajo pametne kartice. Uporaba kubičnih krivulj v namene kriptografije je prvi predlagal Victor S. Miller leta 1985, a so le te v širšo rabo vstopile še le okoli leta 2004.

2 Kubične krivulje

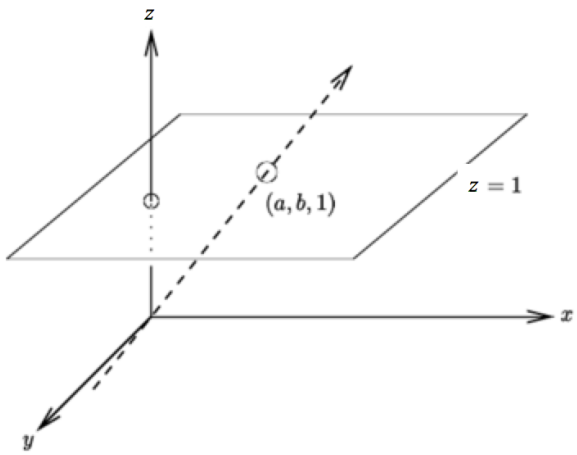
KOPIRANO IZ DIPLOME ALI JE OK????

2.1 Točke na krivulji

Definicija 2.1.

Projektivna ravnina $\mathbb{P}^2$ nad poljem $\mathbb{F}$ je kvocientni prostor $\mathbb{F}^3 - \{0\}/\sim$, kjer je ekvivalenčna relacija podana z $(a, b, c) \sim (\alpha a, \alpha b, \alpha c)$ za vsak $\alpha \in \mathbb{F} \setminus \{0\}$. 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 2.2.

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 2.3.

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 torej oblike

$$a_{300}x^3 + a_{210}x^2y + a_{201}x^2z + a_{120}xy^2 + a_{102}xz^2 + \\ + a_{012}yz^2 + a_{030}y^3 + a_{003}z^3 + a_{111}xyz + a_{021}y^2z = 0,$$

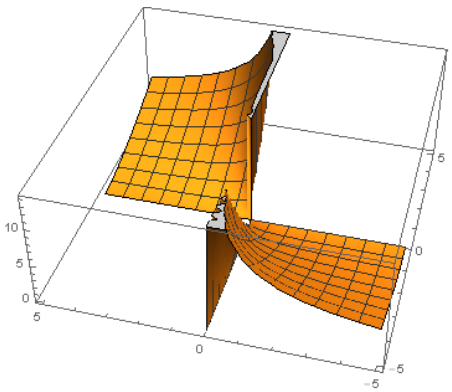
kjer so $a_{ijk} \in \mathbb{F}$. Ta zapis vsebuje 10 koeficientov, vendar se v gladkih primerih lahko polinom poenostavi.

Definicija 2.4.
Algebraična krivulja je *gladka*, če nima nobenih samopresečišč ali singularnosti.

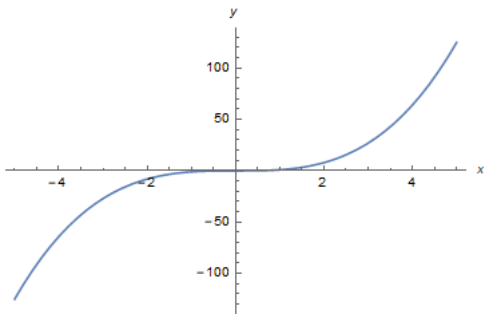
Izrek 2.5 ([?], Izrek 15.2)**.**
Gladko kubično krivuljo nad algebraično zaprtim poljem lahko zapišemo v Weierstrassovi obliki

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

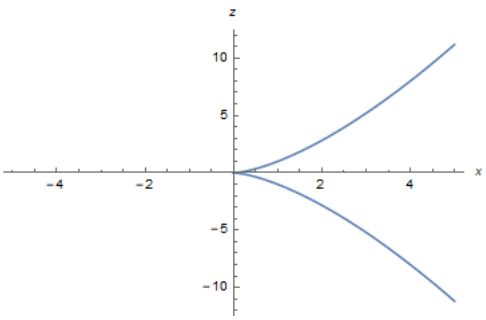
Primer 2.6.
Polinom $P(x, y, z) = z^2y - x^3$ je homogen polinom stopnje 3. Rešitve enačbe $z^2y - x^3 = 0$ pa podajajo točke na kubični krivulji.



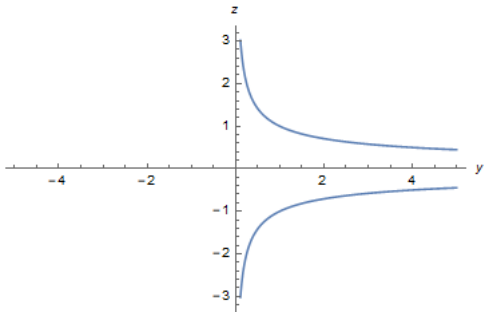
Slika 2: Algebraična krivulja, podana s polinomom $z^2y - x^3$.



Slika 3: Presek algebraične krivulje z ravnino $z = 1$.



Slika 4: Presek algebraične krivulje z ravnino $y = 1$.



Slika 5: Presek algebraične krivulje z ravnino $x = 1$.

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

V nadaljevanju nas bodo zanimale predvsem gladke kubične krivulje v polju $\mathbb{Z}/n\mathbb{Z}$.

Definicija 2.7. Za dani števili a, $b \in \mathbb{Z}/n\mathbb{Z}$ je *kubična krivulja* nad poljem $\mathbb{Z}/n\mathbb{Z}$ množica točk

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

Drugače povedano, afina kubična krivulja je množica rešitev enačbe

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

Pri čemer upoštevamo zvezo med afinimi in projektivnimi koordinatami točk:

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

2.2 Struktura grupe na kubičnih krivuljah

Za definicijo grupe na kubičnih krivuljah uvedimo najprej pomožno operacijo

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

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

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

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

Trditev 2.8.
Operacija $*$ ima naslednje lastnosti:

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

Izrek 2.9.
Kubična krivulja ($\mathcal{C}_P$,+) je Abelova grupa za operacijo

$$\begin{array}{rccc} + : & \mathcal{C}_P \times \mathcal{C}_P & \to & \mathcal{C}_P \\ & (A, B) & \to & (A * B) * O \ , \end{array}$$

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

Dokaz.
S pomočjo trditve 2.8 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 absorbcijo.

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

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

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

Lema 2.10 (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 2.11. Če krivuljo $\mathcal{C}_P$ predstavimo v projektivni ravnini, torej s homogenim polinomom $yz^2 = x^3 + \alpha x^2 z + \beta x z^2 + \gamma y z^2$ je prevoj $O = [0, 1, 0]$.

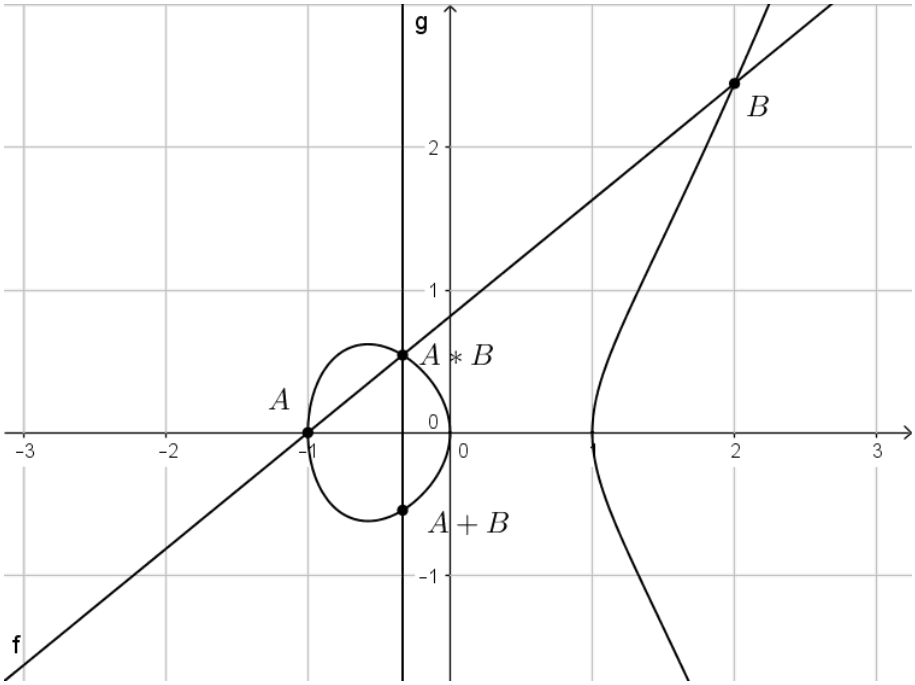
Primer 2.12.
Na spodnji sliki je v afini ravnini $y = 1$ prikazano kako grafično seštevamo točke na Weierstrassovi kubiki $yz^2 - x(x-y)(x+y) = 0$. Sešteti želimo točki $A = (-1, 0)$ in $B = (2, \sqrt{6})$.

Primer 2.13.
Seštejmo točki $A = (-1, 0), B = (2, \sqrt{6})$ na Weierstrassovi kubični krivulji $yz^2 - x(x-y)(x+y) = 0$ v preseku s projektivno ravnino $y = 1$ še računsko z uporabo zgornje leme 2.10. Prepišimo našo krivuljo najprej v afino obliko iz leme 2.10. Ker smo v ravnini $y = 1$, najprej zamenjajmo vlogi y in z.

$$z^2 - x(x-1)(x+1) = 0$$

Dobimo $z^2 = x^3 - x$, torej je $\alpha = 0$, $\beta = -1$ in $\gamma = 0$. Izračunajmo sedaj λ in μ, 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

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

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

3 Diffie-Hellmanova izmenjava ključev nad gladkimi kubičnimi krivuljami

Diffe-Hellmanova izmenjava ključev je postopek, pri katerem se dve osebi npr. Alenka in Boris dogovorita za skrivni ključ na takšen način, 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 1 Diffie-Hellmanova izmenjava ključev.

1. Alenka in Boris se dogovorita za elitpično krivuljo E nad končnim obsegom $\mathbb{F}q$, ter za točko $P \in E(\mathbb{F}_q)$.
2. Alenka se odloči za skrivno število $a \in \mathbb{N}$, in izračuna $P_a = aP$, ter to pošlje Borisu.
3. Boris se odloči za skrivno število $b \in \mathbb{N}$, in izračuna $P_b = bP$, ter to pošlje Alenki.
4. Alenka izračuna $aP_b = abP$.
5. Boris izračuna $bP_a = baP$.

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.

Definicija 3.1. Naj bosta $a, b \in \mathbb{N}$, ter naj bo p praštevilo. Iščemo število k tako, da bo

$$a^k \equiv b \pmod{p}.$$

Trditev 3.2. *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 $a(bP)$.

□

3.1 Index Calculus

Naj bo p praštevilo in naj bo g generator ciklične grupe $\mathbb{F}_p^\times$. Naj $L(h)$ označuje vrednost, za katero velja

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

Iz definicije $L(h)$ sledi, da velja

$$L(h_1 h_2) = L(h_1) + L(h_2) \pmod{p}.$$

Idejo napada na problem diskretnega logaritma v taki grupi najlažje vidimo na primeru.

Primer 3.3. Naj bo $p = 1217$ in $g = 3$. Rešiti hočemo $3^k \equiv 37 \mod 1217$. 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. Išemo x-e tako, da bo

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

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

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

od koder sledi $L(-1) \equiv 608 \mod 1216$. 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 bobimo $L(2) = 216, L(11) = 1059, L(7) = 113, L(5) = 819, L(13) = 87, L(3) = 1$. Sedaj poračunamo za različne j vrednost $3^j * 37$, dokler ne dobimo $3^j * 37 \equiv$ 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 $k = 588$.

3.2 Index Calculus program

```
import mip
import math
import random
import numpy as np
from base import *


def PrimeSearch(num,mod,base):
    """
    Opis:
        PrimeSearch vrne razcep na prafaktorje iz baze ce
        tak razcep obstaja, v nasprotnem primeru vrne None.

     Definicija:
        PrimeSearch(num,mod,base)

     Vhodni podatki:
        num...stevilo, ki ga hocemo razcepiti na prafaktorje
        mod...modul s katerim delamo
        base...baza prastevil s katerimi delamo

     Izhodni podatek:
        seznam_prafaktorjev ,kjer prvi element pomeni
        ali je spredaj - ali + (ce je element 0
        potem je +, ce je element 1 potem je -)
        ce tak razcep obstaja(npr. [1,2,0,3,1] pri bazi
        [2,3,5,7] pomeni da se to stevilo
        zapise kot -2**2*3**0*5**3*7), sicer None
    """

    factors = [0]*len(base)
    num1 = NumberMod(num,mod)
    Prime = num1.isPrime()
    #ce je stevilo prastevilo pogledamo
    #ce lahko razcepimo stevilo - mod
    if Prime and num1.num not in base:
        num1 = NumberMod(-num,mod)

    stevilo = num1.num
    st = 0
    for el in base:
        while stevilo % el == 0:
            factors[st] += 1
            stevilo = stevilo // el
```

```
            st += 1

        if Prime and num1.num not in base and stevilo == 1:
            odg = [1] + factors
        else:
            odg = [0] + factors

        if (odg != [1] +[0]*len(base) and
            odg != [0]*(len(base)+1) and
            stevilo == 1):
            return odg
        else:
            return None

def FindSystemMod(mod,g,base):
    """
    Opis:
       FindSystemMod vrne sistem enacb, ki jih moramo
       resiti, da resimo problem diksretnega logaritma.

     Definicija:
       FindSystemMod(mod,g,base)

     Vhodni podatki:
       g...generator multiplikativne grupe $\Z_mod$.
       mod...modul s katerim delamo
       base...baza prastevil s katerimi delamo

     Izhodni podatek:
       par (A,b)...A je matrika sistema,
                   b je desna stran sistema
    """
    #kratnik doloca koliko enacb je v sistemu
    #(n pomeni dolzina_baze*n enacb)
    kratnik = 5
    #pove ali dodamo enacbo v sistem ali ne
    dodamo = False
    #pove ali ze imamo pokrite vse spremenljivke
    #z dosedanjimi enacbami
    VSE = False
    #porazdelitev spremenljivk v posameznih
    #enacbah ki smo jih dodali
    mamo = [[0]*len(base) for i in range(kratnik*len(base))]
    #katere spremenljivke smo ze pokrili
    mamoSPR = [0]*len(base)
    st = 0#stevilo dodanih enacb
```

```
    A = []
    b = []
    for i in range(1,mod-1):
        #razcepimo stevilo na faktorje iz baze
        odg = PrimeSearch(g**i,mod,base)
        if odg != None:
            #pogledamo ce smo dobili kaksno
            #novo spremenljivko
            if VSE == False:
                for j in range(1,len(base)+1):
                    #zamik zarad +- 1 spredi
                    if odg[j] != 0 and mamoSPR[j-1] == 0:
                        #ce se nimamo pokrite te
                        #spremenljivke to enacbo
                        #vzamemo
                        mamoSPR[j-1] = 1
                        dodamo = True

            elif VSE and len(A) < kratnik*len(base):
                #nimamo se dovolj velikega sistema
                #preverimo ce tako enacbo ze mamo
                dodamo = True
                tmp = list(map(bool,odg[1:]))
                for k in range(st):
                    if mamo[k] == tmp:
                        dodamo = False
            if dodamo:
                mamo[st] = list(map(bool,odg[1:]))
                st += 1
                A.append(odg[1:])
                b.append(i-((mod-1)//2)*odg[0])
                dodamo = False
                if np.prod(mamoSPR) == 1:
                    VSE = True


        if np.prod(mamoSPR) != 0 and len(A) >= kratnik*len(base):
            #prekinemo iskanje ce imamo dovolj enacb
            break


    return(A,b)



def SolveSystemMod(A,b,mod):
```

```
    """
Opis:
    SolveSystemMod vrne resitev sistema linearnih enacb
    podanih z matrikama A in b (Ax = b) modulo stevilo mod.
    Metoda za resevanje sistem pretvori v novo obliko tako,
    da vsako vrstico pretvori v novo enacbo in potem
    celotni sistem resi s pomocjo celostevilskega linearnega
    programa iz paketa mip

    Zgled:
        A = [[0   ,1        ,0        ,0        ,0        ,0        ],
             [2   ,0        ,0        ,1        ,0        ,1        ],
             [0   ,0        ,3        ,0        ,0        ,0        ],
             [1   ,0        ,2        ,0        ,0        ,0        ],
             [0   ,0        ,1        ,0        ,1        ,0        ],
             [0   ,0        ,0        ,0        ,0        ,1        ]]
        b = [1,-584,25,-578,-554,87]

        mod = 1216

        posamezno vrstico pretvori tako, da namesto x2 = 1  dobimo
        vrstico x2 + mod*X1(dodatna spremenljivka) = 1
        (za vsako vrstico uvedemo novo dodatno spremenljivko)

    Definicija:
        SolveSystemMod(A,b,mod)

    Vhodni podatki:
        A...matrika podana kot [[...],[...],[...],...,[...]]
        b...desna stran sistema podana s seznamom
        mod...modulo glede na katerega resujemo sistem

    Izhodni podatek:
        x  seznam vrednosti resitve sistema
    """
    n =len(A)
    dol_vrstice = len(A[0])
    Kopija = A[:]
    #pretvorimo matriko A v primerno obliko
    for i in range(n):
        tmp = [0]*n
        tmp[i] = mod
        Kopija[i] = Kopija[i] + tmp

    #naredimo linearni program(zanimajo nas samo
    #spremenljivke, ki predstavljajo bazo,
```

```
    #katere omejimo med 0 in mod-1)
    m = mip.Model()
    x = [ m.add_var(var_type=mip.INTEGER,lb=1,ub = mod-1)
          for i in range(dol_vrstice) ]
    #spremenljivk tok kukr jih je v vrstici
    for i in range(n):
        x.append(m.add_var(var_type=mip.INTEGER,lb = -10,ub = 10))
        #dodatna spremenljivka za vsako vrstico posebaj
    m.objective = mip.minimize(
        mip.xsum(0*x[j] for j in range(dol_vrstice)))
    for i in range(n):
        m += mip.xsum(
            Kopija[i][j]*x[j] for j in range(dol_vrstice + n)) == b[i]
    m.optimize()

    #zaokrozimo za lepsi izpis(vcasih
    #lahko pride resitev 0.999999999 namesto 1, kar je isto)
    resitev = [int(round(x[i].x)) for i in range(dol_vrstice)]

    return(resitev)


def IndexCalculus(g,mod,a,base):
    """
    Opis:
        IndexCalculus vrne tak $k$, da velja
        $$g^k \equiv a (mod mod)$$, kjer je g generator
        multiplikativne grupe $\Z_mod$.

        Zgled:
            g = 3
            mod = 1217
            a = 37
            base = [2,3,5,7,11,13]

     Definicija:
        IndexCalculus(g,mod,a,base)

     Vhodni podatki:
        g...generator multiplikativne grupe $\Z_mod$.
        mod...modul s katerim delamo
        a...desna stran problema, ki ga resujemo
        base...baza prastevil s katerimi delamo

     Izhodni podatek:
        $k$, da velja $$g^k \equiv a (mod mod)$$
```

```
    """
    Temp = FindSystemMod(mod,g,base)
    A = Temp[0]
    b = Temp[1]
    resitev = SolveSystemMod(A,b,mod-1)
    k = 0

    for i in range(1,mod-1):
        faktorji = PrimeSearch((g**i)*a,mod,base)
        if faktorji != None:
            k = sum([a*b for a,b in zip(faktorji[1:],resitev)])
            k+=faktorji[0]*((mod-1)//2) -i
            return NumberMod(k,mod-1).num


#primer
base = [2,3,5,7,11,13]
g = 27
mod = 1217
k = IndexCalculus(g,mod,37,base)
```

4 Parjenja

Parjenja imajo pomembno vlogo pri napadih na problem diskretnega logaritma nad gladkimi kubičnimi krivuljami.

Definicija 4.1. *Eliptična* krivulja je gladka kubična krivulja.

Definicija 4.2. Naj bo E eliptična krivulja nad poljem K, ter naj bo $n \in \mathbb{N}$. *Torizjske točke* so množica

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

Izrek 4.3. *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*

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

Dokaz. Se ne vem kako bo napisan □

Definicija 4.4. 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 4.5. *γ_n je element $\mathbb{Z}[x, y^2, A, B]$, za vse lihe n. Za sode n pa je γ_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 γ_{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 $n = 2m$.

□

Definicija 4.6. Naj bo K polje in naj bo $n \in \mathbb{N}$ tak, da karakteristika K ne deli n.

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

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

Trditev 4.7. *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*

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

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 nedegenerirana 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 ρ iz $\bar{K}$, za katere je ρ identiteta na koeficientih E.*
- *$e_n(\alpha(S), \alpha(T)) = e_n(S, T)^{deg(\alpha)}$ za vse separabilne endomorfizme α polja E.*

Posledica 4.8. *Naj bosta T_1, T_2 baza $E[n]$. Potem je $e_n(T_1, T_2)$ generator grupe μ_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 4.7 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 4.7 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 valja za vsak S po točki dva trditve 4.7 sledi, da je $dT_2 = \infty$. To pa je mogoče le če $n|d$, kar pomeni da je $n \leq d$. □

5 MOV

MOV napad s pomočjo Weilovega parjenja pretvori problem diskretnega logaritma iz $E(\mathbb{F}q)$ v problem diskretnega logaritma nad $\mathbb{F}_{q^m}^{\times}$. Na ta način se izognemo težji strukturi grupe. Nov problem diskretnega logaritma pa lahko sedaj rešimo z različnimi napadi, med drugim tudi z napadom Index-Calculus 3.1. 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 5.1. *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 4.3 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 4.8 velja $e_N(P, R) = \zeta$ je generator μ_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 $Q = aP$. □

Ideja dokaza nam sedaj, da korake MOV napada.

Algoritem 2 MOV napad

Izberi m tako, da

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

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

1. 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 ζ_1 in ζ_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 $k \mod d$.
6. Ponovi korake 1-5 za različnke točke T dokler ni k določen.

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

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

kjer je krivulja $E(\mathbb{F}_p)$ in MOV pretvori to grupo v $\mathbb{F}_{p^k}^{\times}$.

[2] [1]

Literatura

[1] J. H. Silverman, *The Arithmetic of Elliptic Curves*, Springer, second edition izd., 2009.

[2] L. C. Washington, *Elliptic Curves: Number Theory and Cryptography*, Chapman and Hall/CRC, second edition izd., 2008.