Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Katja Logar

Interaktivna predstavitev računanja na eliptični krivulji

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Aleksandar Jurišić

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Osrednji cilj diplomskega dela naj bo predstavitev aritmetike končnih obsegov in računanja v grupi na eliptični krivulji. V delu vpeljite osnovne kriptografske pojme, kot so npr. enosmerne funkcije in asimetrični kriptosistemi, ki vključujejo dogovor o ključu in digitalne podpise. Pojasnite tudi nekaj napadov na problem diskretnega logaritma, s katerimi določimo minimalno dolžino ključev kriptosistema (glede na kratkoročno oz. dolgoročno varnost) in so v tem smislu temelj za varnost ustreznih kriptositemov. Poudarek naj bo na kriptosistemih, ki so povezani z eliptičnimi krivuljami. Preučite tudi osnove za učinkovite implementacije teh kriptosistemov. Končni cilj naj bo interaktivna internetna aplikacija za aritmetiko na eliptični krivulji nad realnimi števili, kakor tudi praštevilskimi obsegi majhnih moči.

Iskreno se zahvaljujem mentorju, prof. dr. Aleksandru Jurišiću, za strokovno pomoč, nasvete pri izdelavi diplomske naloge ter hitro odzivnost.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
AES	Advanced Encryption Standard	standard za simetrično šifriranje
CSS	Cascading Style Sheets	kaskadne stilske predloge
DLP	Discrete Logarithm Problem	problem diskretnega logaritma
DOM	Document Object Model	objektni model dokumenta
DSA	Digital Signature Algorithm	algoritem za digitalno podpisovanje
EC	Elliptic Curve	eliptična krivulja
ECC	Elliptic Curve Cryptosystem	kriptosistem z eliptičnimi krivuljami
ECDLP	Elliptic Curve Discrete	problem diskretnega logaritma
	Logarithm Problem	nad eliptičnimi krivuljami
ECDSA	Elliptic Curve Digital	algoritem za digitalno podpisovanje
	Signature Algorithm	z eliptičnimi krivuljami
GF	Finite (Galois) Field	končni obseg
HTML	Hypertext Markup Language	jezik za označevanje hiperbesedila
PaaS	Platform as a Service	platforma kot storitev
RSA	Rivest–Shamir–Adleman	RSA kriptosistem z javnimi
	Cryptosystem with Public Keys	ključi

Povzetek

Naslov: Interaktivna predstavitev računanja na eliptični krivulji

Avtor: Katja Logar

Opišemo eliptične krivulje in računanje z njimi. Predstavimo problem diskretnega logaritma in njegovo uporabo v kriptografskih protokolih, kot so Diffie-Helmanov dogovor o ključu ter podpisi DSA in ECDSA. Preučimo najbolj pomembne splošne napade na problem diskretnega logaritma, s poudarkom na tistih, ki delujejo tudi v grupi na eliptični krivulji, to sta algoritem veliki korak–mali korak ter Pollardov ρ-algoritem s Floydovim algoritmom za iskanje ciklov.

Opišemo tudi aplikacijo za interaktivno predstavitev eliptičnih krivulj ter računanje s točkami na eliptični krivulji. Razdeljena je na dve manjši aplikaciji, prva predstavlja eliptične krivulje nad realnimi števili, druga pa nad praštevilskimi obsegi.

Ključne besede: eliptične krivulje, končni obsegi, kriptografija z javnimi ključi.

Abstract

Title: Interactive representation of arithmetic on elliptic curve

Author: Katja Logar

We first describe elliptic curves and their arithmetic. Then we introduce the discrete logarithm problem and its implications in cryptographic protocols, such as the Diffie-Helman key agreement and the signatures DSA and ECDSA. The most important general attacks on the discrete logarithm problem are also studied, with an emphasis on those that work in the group of points on an elliptic curve, i.e., the giant step-baby step algorithm and the Pollard's ρ-algorithm together with the Floyd's algorithm for cycle detection.

Our application for interactive representation of elliptic curves and their arithmetic is also described. It is divided into two smaller applications. The first focuses on elliptic curves over real numbers and the second one on prime fields.

Keywords: Elliptic Curves, Finite Fields, Public Key Cryptography.

Poglavje 1

Uvod

Že v antiki so si želeli ljudje izmenjevati sporočila, ki bi jih znala razumeti zgolj pošiljatelj in prejemnik. S pojavom svetovnega spleta je zahteva po zasebnosti in varnosti postala še toliko bolj pomembna. Ne želimo si namreč, da bi nepooblaščena oseba brala naša zasebna sporočila ali dostopala do podatkov o transakcijah, ko plačujemo vsakodnevne račune.

Na začetku smo si pomagali s simetrično kriptografijo, ki je od prejemnika in pošiljatelja zahtevala, da poznata skupen ključ. Kmalu je prišlo do potrebe, da lahko po spletu varno komuniciramo z drugimi, npr. banko ali strankami, s katerimi se za tak ključ še nismo dogovorili. Alternativo je ponudila asimetrična kriptografija, ki deluje po principu javnih in zasebnih ključev. Za ta namen se danes najpogosteje uporablja kriptosistem RSA. Ker pa je računska moč vsako leto večja, se morajo ustrezno povečevati tudi velikosti ključev (več o izbiri dolžine ključev v Lenstra, Verheul [19]). To je neugodno na napravah, ki imajo omejena sredstva, kot sta energija in računska moč. To so lahko pametne kartice ali pa naprave IoT, katerih število raste veliko hitreje kot pa število klasičnih računalnikov.

Alternativo kriptosistemu RSA nam ponudi kriptosistem z eliptičnimi krivuljami (ECC). Ta je prav tako zasnovan na javnih in zasebnih ključih, a za enako kriptografsko moč oz. varnost potrebuje bistveno krajše ključe. Tako je 256-bitni ključ ECC enakovreden 3072-bitnemu ključu RSA [6], ki ga upo-

rabljamo v naših potnih listih. To pa nam omogoča že bistvene prihranke. Prav tako zaenkrat še ne obstaja algoritem, ki bi prej kot v eksponentnem času rešil problem diskretnega logaritma nad pravilno izbrano eliptično krivuljo, kot navajata Barsagade in Meshram [8].

Cilj naše diplomske naloge torej je, da bi ljudem, ki jih kriptografija zanima, a se še ne spoznajo na eliptične krivulje, čimbolj nazorno predstavili, kaj te sploh so in kako se računa v grupi na eliptični krivulji. Če krivulje rišemo nad obsegom realnih števil, gre za zvezne krivulje, ki pa so lahko sestavljene tudi iz dveh komponent. Pravilo za grupno operacijo se da nad realnimi števili tudi zelo lepo predstaviti s sekantami, tangentami in zrcaljenjem preko abscisne osi. V primeru končnega obsega pa je EC načeloma le končna množica točk in se ne moremo več opreti na geometrijo. Tu nastopi naša aplikacija, ki nam pomaga pri vizualizaciji in računanju. Omogoča, da se uporabniku izrisujejo eliptične krivulje v odvisnosti od parametrov, ki jih izbira. Prav tako so predstavljeni izračuni, ki so potrebni za aritmetiko na krivulji, tako da se lahko uporabnik preizkusi, ali krivulje resnično razume ter nato rezultate preveri ob pomoči aplikacije. Diplomska naloga pa nas bo popeljala tudi do kriptografskih protokolov, ki jih uporabljamo v praksi (DH-dogovora o ključu, digitalnega podpisa z algoritmoma DSA in ECDSA, ...).

V 2. poglavju so predstavljene matematične osnove za razumevanje nadaljnjih poglavij, s poudarkom na grupah in obsegih. 3. poglavje opisuje problem diskretnega logaritma, njegovo uporabo (DH-dogovor o ključu, DSA) ter napade na diskretni logaritem. Tu se predvsem osredotočimo na algoritem veliki korak - mali korak ter Pollardov ρ algoritem s Floydovim algoritmom. V 4. poglavju opišemo eliptične krivulje in računanje s točkami na krivulji. Prav tako je opisan problem diskretnega logaritma, njegova uporaba in možni napadi. 5. poglavje predstavi uporabljeno tehnologijo pri izdelavi aplikacije ter opiše interaktivno predstavitev eliptične krivulje nad realnimi števili ter končnim obsegom $\mathbb{Z}_p$. V zadnjem poglavju je predstavljen zaključek in sklep dela.

Poglavje 2

Končni obsegi

Za razumevanje eliptičnih krivulj nad končnimi obsegi, ki se uporabljajo v kriptografiji, moramo najprej spregovoriti nekaj o samih končnih obsegih, za katere pa je potrebno poznavanje osnov grup in obsegov. To poglavje je povzeto po člankih Jurišić [15] in Tonejc [29].

Definicija 2.1 *Grupa* je algebraična struktura, ki jo predstavimo s parom $(G, \circ)$, kjer je G množica elementov in $\circ$ binarna operacija, z naslednjimi lastnostmi:

- množica G je zaprta za $\circ$, tj. za vse $a, b \in G$ je $a \circ b \in G$,
- velja pravilo o združevanju, tj. za vse $a, b, c \in G$ velja $a \circ (b \circ c) = (a \circ b) \circ c$,
- obstaja element identitete oz. enota $e \in G$, kjer za vsak $a \in G$ velja $a \circ e = a = e \circ a$,
- za vsak $a \in G$ obstaja inverzni element a^{-1}, tj. velja $a \circ a^{-1} = e = a^{-1} \circ a$.

V primeru eliptičnih krivulj bomo imeli opravka z Abelovimi grupami, torej bo veljalo tudi pravilo o zamenjavi: $a \circ b = b \circ a$, za vse $a, b \in G$.

Red elementa grupe $g \in G$ je najmanjše naravno število n, tako da velja $g^n = 1$.

Grupa je *ciklična*, če obstaja tak element a, da velja $G = \{e, a, a^2, \ldots, a^{n-1}\}$ in je $a^n = e$, kjer je $n = |G|$. Rečemo, da a *generira* G oz. da je *generator*.

Definicija 2.2 *Obseg*, ki ga lahko zapišemo kot trojico $(\mathcal{O}, +, *)$, kjer $\mathcal{O}$ prestavlja množico, + in * pa poljubni binarni operaciji na tej množici, je algebraična struktura, za katero veljajo sledeče lastnosti:

- par $(\mathcal{O}, +)$ je grupa z enoto 0,
- par $(\mathcal{O} \setminus \{0\}, *)$ je grupa z enoto 1,
- pravilo o porazdeljevanju, tj. za vse $a, b, c \in \mathcal{O}$ velja:
  $a * (b + c) = a * b + a * c$ in $(b + c) * a = b * a + c * a$.

Obseg je *končen*, če je množica $\mathcal{O}$ končna. Njegovo število elementov oz. *moč* označimo z $|\mathcal{O}|$. Tak končni obseg imenujemo tudi Galoisov obseg, po matematiku Evariste Galoisu, ki je postavil temelje teorije končnih obsegov. Zato končni obseg s q elementi označujemo s kratico GF(q) (angl. *Galois Field*).

Podobseg je podmnožica obsega, ki je za isti operaciji + in * tudi obseg.

Trditev 2.1 *Za vsak končen obseg $\mathcal{O}$ velja, da obstaja najmanjše naravno število n, tako da za poljuben element $a \neq 0$ iz obsega $\mathcal{O}$ velja*

$$\underbrace{a + a + \ldots + a}_{n} = n \cdot a = 0. \tag{2.1}$$

V tem primeru število n imenujemo ***karakteristika obsega****,*

Dokaz. Prepričajmo se najprej, da karakteristika res obstaja. Zaradi zaprtja seštevanja je $1+1 \in \mathcal{O}$, pa tudi $1+1+1 \in \mathcal{O}, \ldots, \underbrace{1 + \cdots + 1}_{m \cdot 1} \in \mathcal{O}$, vendar to še ne pomeni $\mathbb{N} \subseteq \mathcal{O}$, saj bi bilo to v nasprotju z $|\mathcal{O}| < \infty$. Torej obstajata taki najmanjši števili $m, n \in \mathbb{N}_0$, da velja

$$m \cdot 1 = n \cdot 1 \quad \text{in} \quad m < n. \tag{2.2}$$

Z drugimi besedami, ker je obseg končen, bo slej ko prej moralo priti do "trčenja" in m, n sta najmanjši taki števili.
Če enačbi (2.2) na obeh straneh odštejemo 1, dobimo

$$(m-1)\cdot 1 = (n-1)\cdot 1. \tag{2.3}$$

To nas privede v protislovje z začetno predpostavko, da sta m, n najmanjši taki števili, da drži enakost (2.2). Iz tega lahko sklepamo, da je $m = 0$, kar nas privede do enačbe

$$n \cdot 1 = 0. \tag{2.4}$$

Še več, če enačbo (2.4) pomnožimo z 1^{-1}, dobimo, da je $n = 0$. Torej res obstaja karakteristika n in trditev 2.1 drži. □

Trditev 2.2 *Karakteristika končnega obsega $\mathcal{O}$ je praštevilo. Če jo označimo s p, je $\mathbb{Z}_p$ podobseg obsega $\mathcal{O}$.*

Dokaz. Predpostavimo, da karakteristika n ni praštevilo. V tem primeru lahko n zapišemo kot produkt dveh naravnih števil, tj. $n = a \cdot b$, $a, b \in \mathbb{N}$ in $a \neq 1$, $b \neq 1$. Potem lahko zapišemo:

$$n \cdot 1 = a \cdot (b \cdot 1) = 0. \tag{2.5}$$

Ker sta a, b naravni števili manjši od n, sta elementa $\mathcal{O}^*$ in lahko najdemo inverz a^{-1} ter z njim pomnožimo obe strani enačbe (2.5). Tako dobimo $b \cdot 1 = 0$, kar je v protislovju z minimalnostjo števila n in pomeni, da je n res praštevilo. □

Trditev 2.3 *Moč končnega obsega $\mathcal{O}$ je vedno potenca karakteristike, tj. $|\mathcal{O}| = p^n$, kjer je p karakteristika in n neko naravno število.*

Dokaz. Najprej se prepričajmo, da je obseg $\mathcal{O}$ vektorski prostor nad $\mathbb{Z}_p$. Naj bo $V = \mathcal{O}$, množico skalarjev S pa naj predstavljajo elementi podobsega $\mathbb{Z}_p$.

Preveriti moramo zahteve oz. aksiome za vektorski prostor:

$$\begin{aligned} \vec{u}, \vec{v} \in V &\implies \vec{u} + \vec{v} \in V, \\ \vec{u} \in V, s \in S &\implies s \cdot \vec{v} \in V. \end{aligned} \tag{2.6}$$

Prva sledi iz zaprtja aditivne grupe $(\mathcal{O}, +)$, druga pa je posledica zaprtja multiplikativne grupe $(\mathcal{O} \setminus \{0\}, *)$.

Želimo najti bazo prostora V. Za prvi bazni element $\vec{b_1}$ si izberemo poljuben neničelni element. Če $\{s \cdot \vec{b_1}, s \in S\} \neq V$, vzamemo za naslednji bazni element $\vec{b_2}$ poljuben element iz $V \setminus \{s \cdot \vec{b_1}, s \in S\}$. Če $\{s_1 \cdot \vec{b_1} + s_2 \cdot \vec{b_2}, s_1, s_2 \in S\} \neq V$, vzamemo naslednji bazni element $\vec{b_3}$ iz $V \setminus \{s_1 \cdot \vec{b_1} + s_2 \cdot \vec{b_2}, s_1, s_2 \in S\}$, ... Postopek ponavljamo, dokler ne znamo z baznimi vektorji $\vec{b_1}, \vec{b_2}, \ldots$ izraziti prav vsak element V. Ker je naš obseg končen, mora biti tudi naša baza končna. Razsežnost vektorskega prostora V je torej neko naravno število n. Poljuben vektor lahko zapišemo kot:

$$\vec{v} = \sum \vec{b_i} \cdot s_i, \text{ kjer je } s_i \in \mathbb{Z}_p \quad \text{oz.} \quad \vec{v} = (s_1, s_2, \ldots, s_n). \tag{2.7}$$

Ker so $s_1, s_2, \ldots, s_n$ elementi $\mathbb{Z}_p$, imamo za vsak s_i natančno p možnih vrednosti, kar pomeni, da je število elementov obsega $\mathcal{O}$ enako p^n. □

Najpreprostejši končni obseg je GF(p)=$\mathbb{Z}_p$, tj. primer, ko je $n = 1$. Tak obseg, čigar moč je praštevilo, imenujemo tudi *praobseg* in ga sestavljajo elementi množice $\{0, 1, \ldots, p-1\}$. V tem obsegu poteka seštevanje, odštevanje (tj. prištevanje nasprotnega elementa) in množenje po modulu p, deljenje z od nič različnim elementom pa je zgolj množenje z multiplikativnim inverzom, ki ga najučinkoviteje izračunamo z razširjenim Evklidovim algoritmom.

Tudi naša aplikacija za prikaz eliptičnih krivulj se bo zaenkrat omejila na praštevilske obsege $\mathbb{Z}_p$, $p \in \mathbb{P}$, kjer je $\mathbb{P}$ oznaka za množico praštevil.

Poglavje 3

Problem diskretnega logaritma

Za razumevanje problema diskretnega logaritma (DLP, angl. *discrete logarithm problem*) in njegove težavnosti najprej vpeljimo koncept enosmernih funkcij v kriptografiji.

Definicija 3.1 Da je kriptografska funkcija $f(x)$ *enosmerna*, mora veljati sledeče:

- f je javno znana,
- za poljuben x se da učinkovito izračunati funkcijsko vrednost $f(x)$,
- če poznamo $f(x)$, je težko v doglednem času izračunati njeno prasliko, tj. vrednost x.

Lep primer enosmernih funkcij so zgoščevalne funkcije, s katerimi dolgo besedilo zgostimo v kratek niz. To nam pomaga pri digitalnem podpisovanju, saj potem podpišemo samo zgostitev (angl. *hash*) [27].

Na tem mestu je pomembno poudariti, da za vse enosmerne funkcije, ki so trenutno v uporabi, zgolj verjamemo, da je izračun funkcijske vrednosti inverzne funkcije $f^{-1}(y)$ za dano vrednost $y = f(x)$ težek problem, saj to še ni bilo dokazano. Če bi znali dokazati, da P = NP, enosmerne funkcije zagotovo ne obstajajo, vendar tudi v primeru, da P $\neq$ NP, ni povsem jasno, ali obstajajo, kot je navedeno v Impagliazzo in Luby [12].

V kriptografiji javnih ključev se trenutno uporabljajo predvsem enosmerne funkcije s t.i. *zadnjimi vrati* (angl. *backdoor*), kar pomeni, da je izračun $f^{-1}(y)$ za $y = f(x)$ težko, v primeru, da poznaš dodatno skrivnost, pa problem postane učinkovito rešljiv. Te funkcije sta v kriptografijo v 70-ih letih prva vpeljala Diffie in Helman (angl. *trapdoor one-way function*) [10].

Tu je potrebno izpostaviti tri različne sisteme, ki temeljijo na enosmernih funkcijah:

- kriptosistemi faktorizacije celih števil z glavnim predstavnikom RSA: sistem temelji na tem, da je enostavno izračunati produkt $N = pq$, kjer sta p, q veliki in pravilno izbrani praštevili, faktorizacija, tj. iskanje faktorjev p, q za dan produkt N, pa je težek problem,

- kriptosistemi diskretnega logaritma,

- kriptosistemi z eliptičnimi krivuljami.

V nadaljevanju si bomo podrobneje ogledali problem diskretnega logaritma.

3.1 Diskretni logaritem

Definicija 3.2 Naj bo G končna grupa. Za $\alpha, \beta \in G$, kjer je red elementa α enak n, najdi tako število $x \in \{0, 1, ..., n-1\}$, da velja $\alpha^x = \beta$. Število x, če obstaja, imenujemo *diskretni logaritem z osnovo α elementa β.*

Problem diskretnega logaritma oz. DLP (angl. *discrete logarithm problem)* je torej iskanje takega x, ki ustreza zgornji definiciji. Kot smo omenili, je diskretni logaritem enosmerna funkcija, saj znamo potenciranje α^x učinkovito izračunati, računanje logaritma $\log_\alpha \beta$ pa je najverjetneje težko. Za učinkovito potenciranje uporabimo algoritem *kvadriraj in pomnoži.* Trenutno še ne poznamo nobenega polinomskega algoritma za reševanje DLP za pravilno izbrano grupo.

Algoritem kvadriraj in pomnoži

Če želimo izračunati $\alpha^x \bmod p$, zapišemo eksponent x v binarni obliki

$$\sum_{i=0}^{t} x_i \cdot 2^i,\ x_i \in \mathbb{Z}_2. \tag{3.1}$$

Potem izračunamo vse potence α^j, kjer je $j = 2^k$, $k = 0, \ldots, t$ in zmnožimo samo tiste, za katere je $x_k = 1$, tj. v binarnem zapisu eksponenta x je na k-tem mestu enica. Te potence zmnožimo po modulu p in dobimo rezultat $\alpha^x \bmod p$. V tabeli 3.1 je prikazana različica algoritma, pri kateri potence α^{2^k} računamo sproti. Alternativno, če večkrat računamo potence pri isti osnovi α, lahko potence α^{2^k} izračunamo vnaprej ter jih shranimo v tabelo, nato pa v algoritmu željeno potenco samo preberemo iz ustreznega mesta v tabeli. Tak pristop zmanjša čas računanja potence tudi do trikrat.

VHOD: $\alpha \in \mathbb{Z}_p$, $x = \sum_{i=0}^{t} x_i \cdot 2^i$ IZHOD: $\alpha^x \bmod p$
1. $b \leftarrow 1$. Če $k = 0$, vrni 1
2. $A \leftarrow \alpha$
3. Če $k_0 = 1$, $b \leftarrow \alpha$
4. Za vsak $i = 1, \ldots, t$:
4.1: $A \leftarrow \alpha^2 \bmod p$
4.1: Če $k_i = 1, b \leftarrow A * b \bmod p$
5. Vrni b

Tabela 3.1: Algoritem kvadriraj in pomnoži, glej Menezes, Van Oorschot, Vanstone [22, str. 71].

Lahko pa za taisti algoritem uporabimo drugačen pristop. Naj bo $p(\alpha) = \sum_{i=0}^{t} p_i \cdot \alpha^i$ nek polinom. Potem je algoritem za računanje njegove vrednosti, npr. $p(2)$ naslednji:

$$(\ldots((p_n \cdot 2 + p_{n-1}) \cdot 2 + p_{n-2}) \cdot 2 \ldots) \cdot 2 + p_0 \tag{3.2}$$

Ta algoritem poznamo pod imenom *Hornerjev algoritem*. Naš eksponent x je v resnici vrednost polinoma za $\alpha = 2$, vrednosti p_i pa binarne vrednosti x

na i-tem mestu v binarnem zapisu. Sedaj razširimo računanje te vrednosti na računanje potence α^x, ki jo lahko izračunamo kot:

$$\alpha^{((2+p_{n-1})\cdot 2+p_{n-2})\cdot 2\ldots} \quad \text{oz.} \quad ((\alpha^2 \cdot \alpha^{p_{n-1}})^2 \cdot \alpha^{p_{n-2}})^2 \cdot \ldots \tag{3.3}$$

Opomba: ker je vodilni koeficient p_n vedno enak 1, smo ga v enačbi (3.3) spustili.

V primeru, da algoritem kvadriraj in pomnoži implementiramo preko Hornerjevega algoritma, so možne še nadaljnje optimizacije hitrosti, kot so metode z drsečim oknom, NAF, ... Več o tem v Nose [24].

Za kriptografski sistem na podlagi diskretnega algoritma bi lahko v teoriji uporabili katerokoli končno grupo, a se v praksi izkaže, da so nekatere grupe varnejše od drugih oz. za nekatere poznamo polinomske algoritme za reševanje diskretnega logaritma. Teh se moramo paziti, saj bi preveč povečale dolžino ključev, npr v $(\mathbb{Z}_p, +)$ rešimo DLP z razširjenim Evklidovim algoritmom v kubičnem času. V uporabi so predvsem naslednje grupe [30]:

- multiplikativna grupa končnega obsega GF(q), kjer je q praštevilo ali potenca števila 2,
- grupa na eliptični krivulji nad končnim obsegom,
- razredna grupa nad kvadratnim številskim obsegom,
- deliteljska razredna grupa hipereliptične krivulje malega rodu.

V nadaljevanju se bomo bolj posvetili multiplikativni grupi nad praštevilskim obsegom GF(p)*, tj. $\mathbb{Z}_p^*$, kjer je p praštevilo, in grupi na eliptični krivulji.

3.2 DLP v multiplikativni grupi $\mathbb{Z}_p^*$

Prvo uporabo enosmernih funkcij na osnovi DLP v kriptografiji sta Diffie in Helman leta 1976 predlagala ravno nad multiplikativno grupo praštevilskega obsega $\mathbb{Z}_p^*$, ki ga sestavljajo vsi neničelni elementi $\mathbb{Z}_p$. Ta grupa je ciklična, glej Tonejc [29].

3.2.1 Diffie-Helmanov dogovor o ključu

Diffie-Helmanov dogovor o ključu (povzet po Van Tilborg in Jajodia [30]) predvideva, da sta praštevilo p in generator α multiplikativne grupe $\mathbb{Z}_p^*$, kjer je $\alpha^{p-1} \equiv 1 \pmod{p}$, javno znana. Če se torej Anita in Bojan dogovarjata o ključu, je potek izmenjave sledeč:

- Anita si izbere poljubno naravno število x, kjer je $x \in \{1, \ldots, p-2\}$, izračuna $z_x = \alpha^x \bmod p$ in število z_x pošlje Bojanu.
- Sočasno si Bojan izbere poljubno naravno število y, kjer je $y \in \{1, \ldots, p-2\}$, izračuna $z_y = \alpha^y \bmod p$ in število z_y pošlje Aniti.
- Anita izračuna $s_A = {z_y}^x$ in Bojan izračuna $s_B = {z_x}^y$.

Na koncu izmenjave ključev imata tako Anita in Bojan enak ključ ($s_A = s_B$), ki ga lahko uporabljata za nadaljnjo komunikacijo s poljubnim simetričnim kriptosistemom. Če bi napadalec torej želel odšifrirati nadaljnjo komunikacijo med Anito in Bojanom, bi moral iz α, p, z_x, z_y izračunati α^{xy}. To imenujemo *DH-problem*. Najboljša poznana rešitev tega problema je, da z diskretnim logaritmom iz z_x ugotovimo x ali pa iz z_y ugotovimo y ter se nato postavimo v vlogo Anite oz. Bojana. (Opomba: tak dogovor o ključu ni odporen na napad s posrednikom [30], zato v praksi dodamo še digitalna potrdila.)

3.2.2 Algoritem za digitalno podpisovanje

Spregovorimo najprej o digitalnem podpisovanju. *Digitalni podpis* je kriptografski protokol, s katerim zagotavljamo *pristnost*, *nezatajljivost* in *celovitost* digitalnega dokumenta. Pristnost nam zagotavlja, da je dokument podpisal lastnik ustreznega zasebnega ključa, nezatajljivost pomeni, da pošiljatelj ne more zanikati, da je dokument podpisal, celovitost pa, da podpisani dokument ni bil spremenjen [26].

Digitalni podpis najlažje izvedemo s kriptografijo javnih ključev, kjer z različnimi algoritmi generiramo zasebni in javni ključ. Z zasebnim ključem

lahko dokument podpišemo (oz. v resnici podpišemo zgostitev dokumenta), prejemnik pa lahko potem z našim javnim ključem preveri, ali smo res dokument podpisali mi. Eden od takih algoritmov je tudi DSA.

Algoritem za digitalno podpisovanje DSA (angl. *Digital Signature Algorithm* je bil prva shema za digitalno podpisovanje, ki mu je vlada dala zakonsko veljavo, in sicer se je to zgodilo v Združenih državah Amerike leta 1993, ko je algoritem postal standard (angl. *U.S. Federal Information Processing Standard* ali FIPS). Algoritem je različica ElGamalove sheme za digitalni podpis.

Algoritem lahko razdelimo na tri sklope: generiranje ključa, podpisovanje in preverjanje podpisa. V nadaljevanju bomo vsakega natančneje opisali (Jurišić in Menezes [13] ter FIPS 186-4 [25]).

Generiranje ključa za DSA

Če želi Anita izračunati ključ, mora storiti naslednje:

1. Izbere par (L, N), ki predstavlja bitne dolžine števil p in q. FIPS 186-4 specificira, da je par

$$(L, N) \in \{(1024, 160), (2048, 224), (2048, 256), (3072, 256)\}.$$

2. Izbere N-bitno praštevilo q.
   Izrek o gostoti praštevil pravi, da je N-bitnih praštevil približno $\frac{N}{\ln N}$. Tako naključno izberemo liho N-bitno število ter preverimo, ali je izbrano število res praštevilo. Najbolj znana metoda za preverjanje, ali je število praštevilo, je Erastotenovo sito, ki pa v našem primeru ne pride v poštev, saj je za velika števila prepočasna (časovna zahtevnost $\mathrm{O}(n)$, kar je eksponentna zahtevnost glede na dolžino zapisa števila n). Zato se raje poslužujemo probabilističnih metod, ki nam z dovolj veliko verjetnostjo zagotavljajo, da je izbrano število praštevilo. Dve taki metodi sta Rabin-Millerjeva ter Solovay-Strassenova.

3. Izbere L-bitno praštevilo p, da velja $q \mid p - 1$.
   To naredimo tako, da izberemo poljubno $(L-N)$-bitno število ℓ, tako da

velja $p = q\ell + 1$. Potem enako kot v prejšnjem koraku preverimo, ali je p praštevilo. Če ni, si izberemo novo število ℓ ter postopek ponavljamo, dokler ne najdemo ustreznega praštevila p.

4. Izbere element $h \in \mathbb{Z}_p^*$ in izračuna $g = h^{(p-1)/q} \bmod p$. V primeru, da je $g = 1$, izračun ponovi, dokler ne dobi $g \neq 1$.
   Element g je generator za ciklično grupo moči q nad $\mathbb{Z}_p^*$, saj iz malega Fermatovega izreka sledi $g^q \equiv h^{p-1} \equiv 1 \bmod p$.

5. Izbere naključno naravno število x, z intervala $[1, q-1]$.

6. Izračuna $y = g^x \bmod p$.

7. Parametri algoritma so torej (p, q, g), Anitin javni ključ je element y iz $\mathbb{Z}_p^*$, njen zasebni ključ pa je x.

Podpisovanje

Če želi Anita podpisati sporočilo m, mora storiti naslednje:

1. Izbere naključno naravno število k z intervala $[1, q-1]$.

2. Izračuna $r = (g^k \bmod p) \bmod q$.

3. Izračuna $k^{-1} \bmod q$.

4. Izračuna $s = k^{-1}(H(m) + xr) \bmod q$, kjer je H zgoščevalna funkcija. Originalno je bil v uporabi algoritem SHA-1, danes pa je že v veljavi varnejši SHA-2 ali celo SHA-3.

5. Če je $s = 0$, potem se vrne na korak 1.

6. Podpis sporočila m je par (r, s).

Dejanski podpis dokumenta je torej dolg zgolj $2N$ bitov (v praksi si želimo imeti čim krajše podpise). Zakaj je to potem varno, če računamo v tako majhni grupi $(q-1)$? Odgovor se skriva v tem, da je grupa $\mathbb{Z}_q^*$ podgrupa $\mathbb{Z}_p^*$. Medtem ko znamo $\mathbb{Z}_p^*$ teoretično prevesti na grupo $(\mathbb{Z}_{p-1}, +)$ (grupi sta

izomorfni), pa ne poznamo preslikave iz $\mathbb{Z}_q^*$ v $(\mathbb{Z}_{q-1}, +)$. Zato zaenkrat ne moremo uporabljati algoritmov, ki izkoriščajo strukturo grupe $(\mathbb{Z}_{q-1}, +)$, za lažji izračun DLP, saj ne poznamo neposredne strukture $\mathbb{Z}_q^*$, pač pa le prek $\mathbb{Z}_p^*$.

Na ta način je podpis pri DSA enako dolg kot pri ECDSA, ki ga bomo spoznali v nadaljevanju, ima pa ECDSA še vedno to prednost, da je zaradi krajših ključev računanje hitrejše kot pri DSA.

Preverjanje podpisa

Če želi Bojan preveriti Anitin podpis sporočila m, tj. (r, s), mora storiti sledeče:

1. Na voljo mora imeti parametre (p, q, g), prav tako mora dobiti Anitin javni ključ y.

2. Izračuna $w = s^{-1} \bmod q$ in $H(m)$.

3. Izračuna $u_1 = H(m)w \bmod q$ in $u_2 = rw \bmod q$.

4. Izračuna $v = (g^{u_1} y^{u_2} \bmod p) \bmod q$.

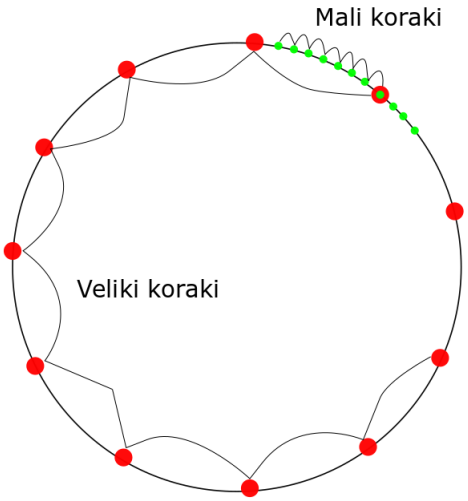
5. Podpis sprejme, če in samo če je $v = r$.
   Preverimo, ali je to res pravi način za preverjanje. Želimo pokazati, da je $g^x = g^{u_1} y^{u_2}$. Če torej v izraz $g^{u_1} y^{u_2}$ vstavimo ustrezne spremenljivke za y, u_1, u_2, dobimo $g^{H(m)w+xrw}$. V eksponentu lahko izpostavimo w ter enačbo preoblikujemo v $g^{k(H(m)+xr)/(H(m)+xr)}$. Eksponent lahko okrajšamo in dobimo g^x. Torej preverjanje podpisa deluje.

3.3 Napadi na DLP

Diskretni logaritem je v praštevilskih obsegih sledeč: za dana elementa $g \in \mathbb{Z}_p^*$ reda $p - 1$ in $h \in \mathbb{Z}_p^*$ poišči tako število $x \in \{0, 1, \ldots, p - 2\}$, da velja $g^x = h$ (mod p). Tak x imenujemo diskretni logaritem h pri osnovi g.

Požrešna metoda (angl. *exhaustive search*) za izračun diskretnega logaritma bi bila, da preizkusimo zaporedoma vse potencialne vrednosti $x \in \{1, 2, \ldots, p-2\}$, vse dokler ne najdemo prave. To je v praksi zelo zamudno (časovna zahtevnost je $O(p)$), saj v kriptografiji izbiramo za p eksponentno velike vrednosti, tj. $p \approx 2^k$, kjer je k velikost zapisa števila p. Obstajajo pa različni algoritmi za reševanje problema diskretnega logaritma, ki so časovno učinkovitejši od požrešnega pristopa. V nadaljevanju bomo predstavili Shankov algoritem veliki korak - mali korak in Pollardov ρ algoritem.

3.3.1 Algoritem veliki korak - mali korak



Slika 3.1: Predstavitev malih in velikih korakov na krogu.

Iščemo torej tak x, da velja

$$g^x \equiv h \pmod{p}. \tag{3.4}$$

Ker mora biti $x \in \{1, \ldots, p-2\}$, je število možnih rešitev končno in glede na to, da računamo po modulu p, si lahko predstavljamo, da vse možne potence $g^0, g^1, \ldots, g^{p-2}$ pripnemo zaporedoma na krog. Ideja algoritma je, da

si vnaprej sestavimo tabelo primerno velikih (o tem več spodaj) izračunanih vrednosti na krogu, potem pa poskušamo z majhnimi koraki "ujeti" eno od naračunanih vrednosti. Ideja algoritma je prikazana na sliki 3.1.

Formalno algoritem poteka sledeče (povzeto po Corot, Lefranc in Poupard [9]):

- Najprej si izberemo tako število m, da velja: $m = \lceil\sqrt{(p-1)}\rceil$. Potem lahko iskano število x zapišemo kot $x = i \cdot m + j$, kjer sta $i, j \in \{0, 1, \ldots, m-1\}$.

- Če tak x vstavimo v prvotno enačbo 3.4, dobimo: $h \equiv g^{im+j} \pmod{p}$. Dobljeno enačbo lahko preoblikujemo:

$$g^{im} \equiv h * g^{-j} \pmod{p}. \tag{3.5}$$

  Levi del enačbe (3.5) nam predstavlja način izračuna velikih korakov, desni pa majhnih.

- Iz izračunanih velikih korakov $h * g^{im} \bmod p$, za $i = 0, 1, \ldots, m-1$ sestavimo tabelo.

- Zaporedoma izračunamo $h * g^{-j} \bmod p$ za $j = 0, 1, \ldots, m-1$ in na vsakem koraku preverimo, ali se izračunana vrednost nahaja v tabeli velikih korakov. V primeru, da najdemo ustrezno vrednost, eksponent x izračunamo kot $x = i \cdot m + j$.

To ni edina možnost implementacije tega algoritma, lahko bi tudi tabelirali male korake in potem računali velike ter primerjali s tabelo malih korakov.

V primeru, da za izračun diskretnega logaritma uporabimo algoritem veliki in mali korak, se nam časovna zahtevnost požrešne metode zmanjša na $O(\sqrt{p})$. Če imamo več časa za predpripravo tabele, bomo še hitreje prišli do trčenja. Ni pa algoritem najbolj prostorsko optimalen, saj je njegova prostorska zahtevnost prav tako $O(\sqrt{p})$, odvisna je od velikosti izračunane tabele. Lahko si namreč vnaprej izračunamo večjo tabelo, da hitreje naletimo na

trčenje. Seveda je to smiselno, če jo bomo večkrat uporabili. Tu lahko vidimo kompromis med časom izvajanja in porabljenim prostorom (večja kot je tabela, manj je računanja).

3.3.2 Pollardov ρ algoritem s Floydovim algoritmom za iskanje ciklov

Pollardov ρ algoritem ima enako kot algoritem veliki korak - mali korak časovno zahtevnost $O(\sqrt{p})$, njegova prostorska zahtevnost pa je, če uporabimo Floydov algoritem za iskanje ciklov, konstantna ($O(1)$). Podpoglavje 3.3.2 je povzeto po Schlegel [28].

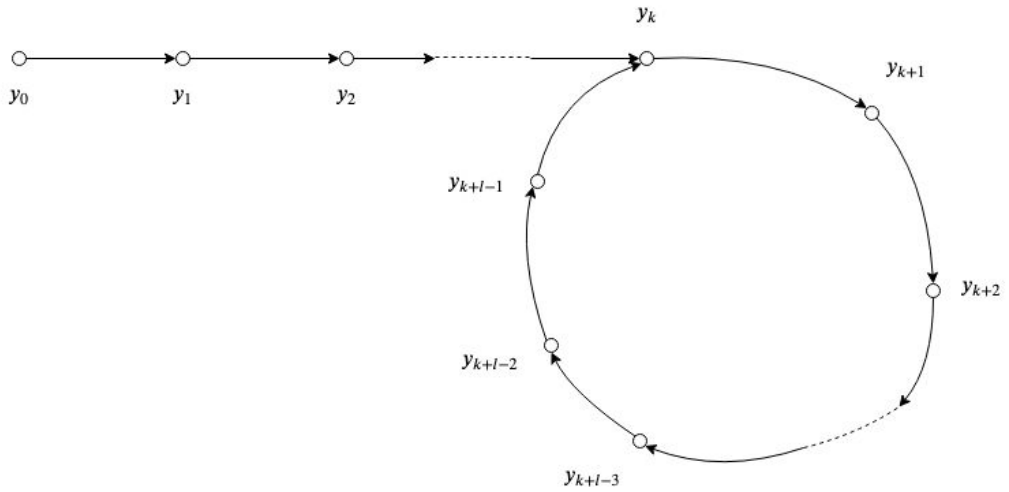
Naj bo G neka končna množica. Na njej definirajmo funkcijo $f : G \to G$ in naključno začetno vrednost $y_0 \in G$. Funkcija f nam naključno izbere nov element iz množice G na podlagi prejšnje vrednosti funkcije. Na ta način gradimo zaporedje $\{y_i\}$, kjer je $y_i = f(y_{i-1})$.

Definicija 3.3 Zaporedje je *periodično*, če obstajata taki števili $m > 0$ in $n_0 \geq 0$, da velja $y_{n+m} = y_n$ za vsak $n \geq n_0$. Število m imenujemo *perioda* zaporedja, n_0 pa *predperioda*. Če sta m in n_0 najmanjši taki števili, pravimo da sta *osnovna perioda* in *osnovna predperioda*.

Ker je množica G končna, je zaporedje $\{y_i\}$ periodično. To pomeni, da obstajata taki naravni števili i in j, da velja $y_i = y_j$ in $i \neq j$. Pravimo, da pride do *trčenja*.

Označimo osnovno periodo z ℓ in osnovno predperiodo s k. Zaporedje $\{y_i\}$ lahko razdelimo na dva dela: *rep*, ki ga sestavljajo elementi $\{y_0, y_1, \ldots, y_k\}$, in *cikel*, ki ga sestavljajo elementi $\{y_k, y_{k+1}, \ldots, y_{k+\ell-1}\}$. Po tej predstavitvi je algoritem tudi dobil ime, saj nas grafična predstavitev zaporedja, ki jo lahko vidimo na sliki 3.2, spominja na grško črko ρ.

Algoritem potem izgleda sledeče: s funkcijo f računamo funkcijske vrednosti in jih shranjujemo v tabelo. Ob vsakem izračunu f preverimo, ali se vrednost funkcije že nahaja v tabeli. Če to ne drži, postopek ponavljamo



Slika 3.2: Periodično zaporedje $\{y_i\}$ z repom dolžine k in ciklom dolžine ℓ

dokler ne pride do trčenja. Kot smo že omenili, do trčenja zagotovo pride, ker je množica G končna.

Zdaj se lahko vprašamo v koliko korakih lahko pričakujemo, da bomo našli trčenje. S probabilističnimi metodami lahko dokažemo, da bomo trčenje z verjetnostjo višjo od 0.5 našli v največ $O(\sqrt{|G|})$ korakih. Dokaz se nahaja v Schlegel [28].

Pollardov ρ algoritem ima torej enako časovno zahtevnost kot algoritem veliki korak - mali korak. Prav tako ima enako prostorsko zahtevnost, tj. $O(\sqrt{|G|})$, saj moramo shranjevati vse člene zaporedja. Prostorsko zahtevnost pa lahko izboljšamo s Floydovim algoritmom.

Pri Floydovem algoritmu iščemo tak najmanjši indeks j, da velja

$$y_j = y_{2j}. \tag{3.6}$$

Sproti torej računamo $\{y_j\}$ in $\{y_{2j}\}$ vse dokler ne drži enačba (3.6). Tako se na vsakem koraku algoritma razlika med številoma konstantno povečuje, dokler ni enaka ravno l oz. naši osnovni periodi. V primeru, da člene zaporedja računamo s Floydovim algoritmom, ne potrebujemo več v spominu hraniti

celotnega zaporedja, temveč samo prejšnja dva člena. Prostorska zahtevnost tako postane konstantna.

Pri našem opisu Pollardovega ρ algoritma smo spustili, kakšna točno mora biti naključna funkcija f. V praksi je v resnici deterministična, a vseeno izgleda naključno oz. ima nekatere lastnosti naključnih funkcij. Več o tem je opisano v Schlegel [28].

Za reševanje problema diskretnega logaritma nad praštevilskim obsegom obstajajo še optimalnejši algoritmi, ki pa ne delujejo pri izračunu diskretnega logaritma nad grupo na eliptični krivulji, zato se v njih ne bomo spuščali.

Poglavje 4

Eliptične krivulje

Prvo omembo eliptičnih krivulj najdemo v antiki, in sicer se je z njimi že v 2. stoletju ukvarjal Diofant. Kasneje so jih preučevali v teoriji števil. Do leta 1920 so se z eliptičnimi krivuljami ukvarjali predvsem Cauchy, Lucas, Sylvester in Poincare. K popularizaciji eliptičnih krivulj je leta 1984 prispeval Lenstra, ki je razvil podeksponentni algoritem za faktorizacijo celih števil [20]. Prav tako je k temu prispevalo dejstvo, da so z uporabo eliptičnih krivulj dokazali zadnji Fermatov teorem in splošni zakon o vzajemnem učinku (povzeto po Barsagade, Meshram [8]).

V kriptografiji so se eliptične krivulje pojavile 1985, ko sta N. Koblitz [17] in V. Miller [23] neodvisno predlagala, da bi pri izračunu diskretnega logaritma uporabili grupo na eliptični krivulji nad končnim obsegom. Trenutno še ne poznamo učinkovitih algoritmov, s katerimi bi bili sposobni izračunati diskretni logaritem nad tako izbrano grupo. Zato so kriptosistemi, ki uporabljajo eliptične krivulje, še posebej primerni za tako ali drugače omejene sisteme, saj nam ECC prinaša manjše ključe, prihranke pri pasovni širini in hitrejšo implementacijo, kot je opisano v Koblitz, Koblitz in Menezes [16]. V tabeli 4.1 so prikazane dolžine ključev pri različnih kriptosistemih za dosego enakega nivoja varnosti [14].

Poglavje je razdeljeno takole: v prvem podpoglavju so opisane eliptične krivulje nad realnimi števili ter pripadajoči grupni zakon, v drugem podpo-

glavju eliptične krivulje nad $\mathbb{Z}_p$, v zadnjem podpoglavju pa je predstavljen problem diskretnega logaritma nad EC. Ta se deli še na Diffie-Helmanov dogovor o ključu, algoritem ECDSA in pa napade na ECDLP, s poudarkom na algoritmu veliki korak - mali korak.

simetrične šifre (AES)	asimetrične šifre (RSA, DSA)	eliptične krivulje (ECDSA)
40 bitov	274 bitov	80 bitov
56 bitov	384 bitov	106 bitov
64 bitov	512 bitov	132 bitov
80 bitov	1024 bitov	160 bitov
112 bitov	2048 bitov	237 bitov
128 bitov	3072 bitov	283 bitov
192 bitov	8192 bitov	409 bitov
256 bitov	16384 bitov	540 bitov

Tabela 4.1: Tabela prikazuje dolžine ključev pri različnih kriptosistemih za enak nivo varnosti (Jurišić, Tonejc [14]).

4.1 Eliptične krivulje nad realnimi števili

Splošna enačba za eliptično krivuljo je sledeča:

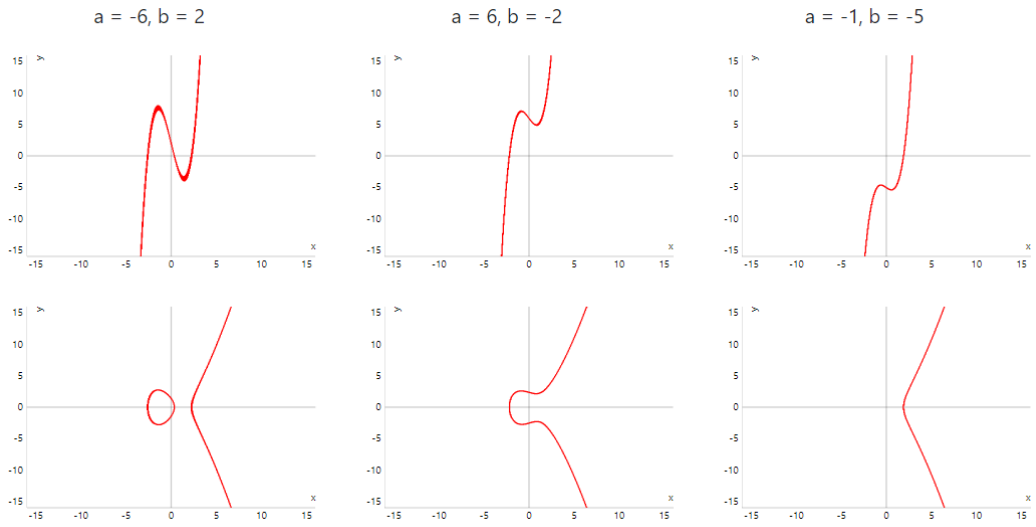
$$y^2 + a_1xy + a_2y = x^3 + a_3x^2 + a_4x + a_5, \tag{4.1}$$

kjer so $a_1, a_2, \ldots, a_5$ realna števila. Če privzamemo, da karakteristika ni enaka 2 ali 3, lahko enačbo eliptične krivulje z uporabo novih spremenljivk zapišemo v Weierstrassovi normalni obliki, ki jo bomo uporabljali tudi v nadaljevanju:

$$y^2 = x^3 + ax + b, \tag{4.2}$$

kjer sta a in b realni števili. Eliptično krivuljo E sestavljajo vse točke (x, y), ki ustrezajo enačbi (4.2) in točka v neskončnosti (∞, ∞), ki jo označimo z 0, kot se za grupo v aditivni obliki spodobi.

V odvisnosti od števila ničel polinoma tretje stopnje ter lokalnih ekstremov, ki se lahko nahajajo nad ali pod abscisno osjo, poznamo tri različne oblike eliptične krivulje. Polinomi tretje stopnje in pripadajoče eliptične krivulje se predstavljeni na sliki 4.1.



Slika 4.1: Polinomi tretje stopnje in pripadajoče eliptične krivulje, pri čemer nas polinom pod abscisno osjo ne zanima, ker je desna stran enačbe (y^2) vedno nenegativna.

4.1.1 Grupni zakon

Za točko $T = (x, y) \in E$ je točka $-T = (x, -y)$ očitno tudi na krivulji E. Za različni točki $P = (x_1, y_1)$ in $Q = (x_2, y_2)$ krivulje E je

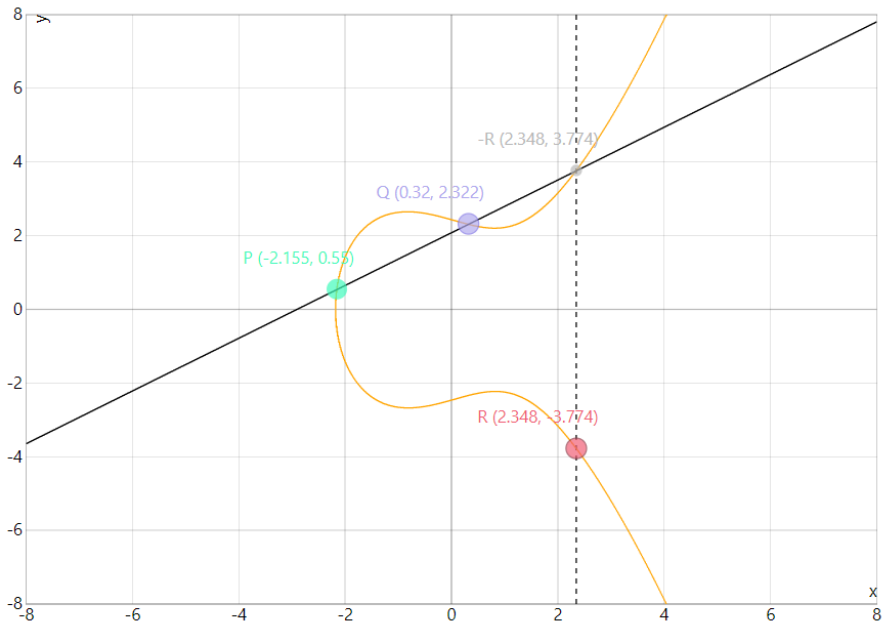
$$R := P + Q \tag{4.3}$$

točka krivulje E, ki jo dobimo na naslednji način. Sekanta skozi točki P in Q je določena z enačbo

$$y = \frac{y_1 - y_2}{x_1 - x_2}(x - x_1) + y_1, \tag{4.4}$$

če $Q \neq -P$, tj. $x_1 \neq x_2$. Če vrednost y iz enačbe (4.4) vstavimo v enačbo (4.2), dobimo kubično enačbo za x, katere dve rešitvi sta x_1 in x_2. Torej mora obstajati tudi tretja realna rešitev x_3, kar pomeni, da omenjena sekanta seka krivuljo še v tretji točki. Če to točko prezrcalimo preko abscisne osi, dobimo točko R. Z drugimi besedami je tretje presečišče točka $-R$. Če je $Q = -P$, je $R = 0$, in če je $Q = 0$, je $R = P$.

Če pa je $P = Q$, nadomestimo sekanto skozi točki P in Q s tangento na krivuljo E v točki P. Torej je 0 enota, $-P$ pa inverz točke P za operacijo $+$, kar pomeni, da sta bili oznaki za (∞, ∞) in $(x, -y)$ smiselni.



Slika 4.2: Seštevanje na eliptični krivulji $y^2 = x^3 - 2x + 6$.

Če torej $P, Q \neq 0$ in $Q \neq -P$ (primer na sliki 4.2) izračunamo koordinate točke R kot:

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2, \\ y_3 &= \lambda(x_1 - x_3) - y_1, \end{aligned} \tag{4.5}$$

pri čemer je λ smernostni koeficient sekante skozi točki P in Q, kadar je

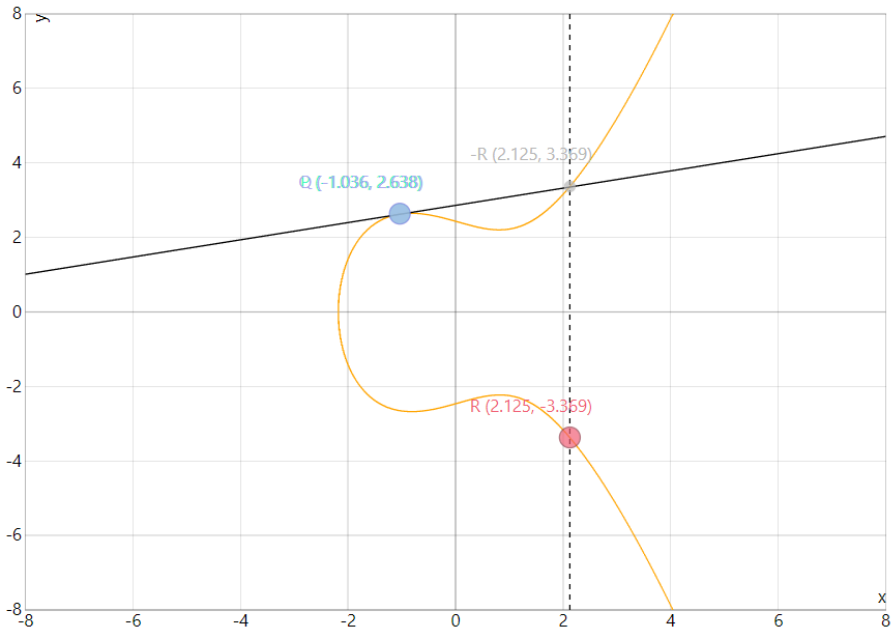
$P \neq Q$ oz. tangente v točki P na krivuljo E, ko je $P = Q$, tj.

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, & \text{če je } P \neq Q, \\ \\ \dfrac{3x_1^2 + a}{2y_1}, & \text{če je } P = Q. \end{cases} \tag{4.6}$$

Z nekaj računanja se lahko prepričamo, da je $(E, +)$ komutativna grupa, čeprav je za dokaz asociativnosti potrebno že več teorije, glej npr. Friedl [11].

Z grupnim zakonom definiramo tudi *skalarno množenje*. Velja, da je množenje s skalarjem enostavno prištevanje točke:

$$nP = \underbrace{P + P + \cdots + P}_{n}. \tag{4.7}$$



Slika 4.3: Podvojitev točke P, tj. $2P$, na eliptični krivulji $y^2 = x^3 - 2x + 6$

Produkt nP lahko učinkovito izračunamo z algoritmom *podvoji in prištej* (tj. aditivna varianta metode *kvadriraj in pomnoži*), ki kompleksnost naivnega seštevanja z $\mathrm{O}(n)$ zmanjša na $\mathrm{O}(\log(n))$.

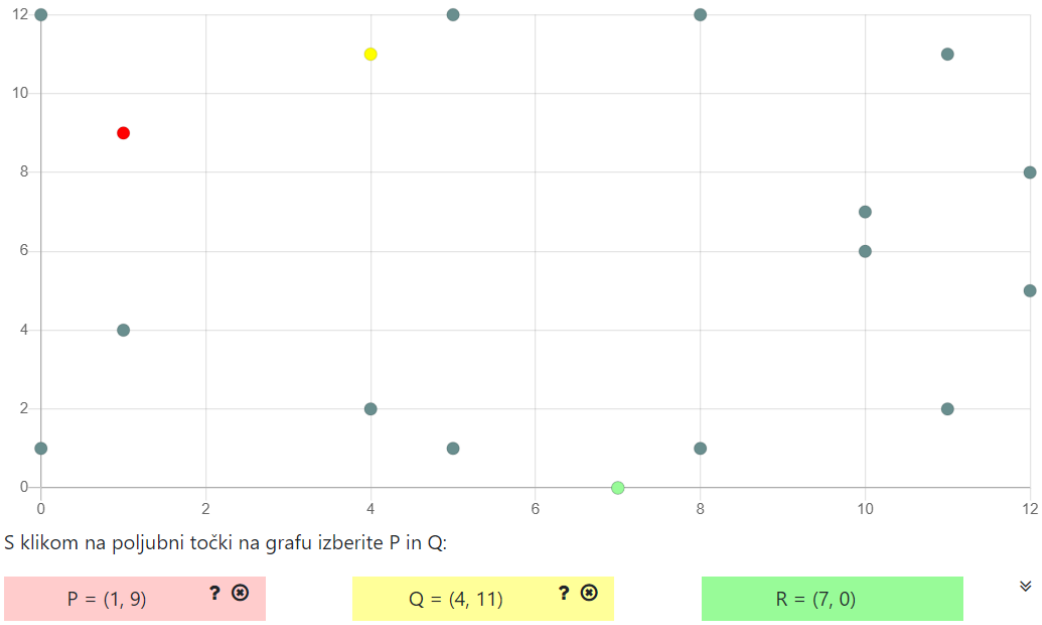
4.2 Eliptične krivulje nad $\mathbb{Z}_p$

Eliptično krivuljo nad $\mathbb{Z}_p$, kar zapišemo na kratko kot $E(\mathbb{Z}_p)$, sestavljajo vse točke (x, y), ki ustrezajo enačbi:

$$y^2 = x^3 + ax + b, \tag{4.8}$$

pri čemer sta a in b elementa $\mathbb{Z}_p$, prav tako pa tudi točka v neskončnosti, ki jo označimo z 0, saj ima vlogo aditivne enote, kot bomo kmalu videli.

Prav tako kot pri eliptičnih krivuljah nad realnimi števili tudi tukaj velja grupni zakon, predstavljen v enačbah (4.3), (4.5) in (4.6), in skalarno množenje (4.7) s to razliko, da v tem primeru vse računamo po modulu p. Primer grupnega zakona nad praštevilskim obsegom je prikazan na sliki 4.4.



Slika 4.4: Grupni zakon na eliptični krivulji $y^2 = x^3 + x + 1$ v $\mathbb{Z}_{13}$.

Podpoglavji 4.1 in 4.2 sta povzeti po Jurišić, Menezes [13].

4.3 Problem diskretnega logaritma nad EC

Za razumevanje diskretnega logaritma nad eliptičnimi krivuljami (ECDLP) potrebujemo še nekaj dodatnih definicij.

Red točke na eliptični krivulji je enak najmanjšemu naravnemu številu n, da velja: $nP = 0$. To je enako kot je definirano v poglavju 2, le da tu velja aditivna notacija.

Diskretni logaritem nad eliptično krivuljo $E(\mathbb{Z}_p)$ je sledeč: Za $P, Q \in E(\mathbb{Z}_p)$ je diskretni logaritem točke Q pri osnovi P enak najmanjšemu naravnemu številu n, za katerega velja:

$$Q = nP. \tag{4.9}$$

Z drugimi besedami, n je diskretni logaritem točke Q pri osnovi P. Problem diskretnega logaritma je torej iskanje najmanjše take vrednosti n, da velja (4.9).

Da je izračun diskretnega logaritma dejansko težek, moramo pravilno izbrati eliptično krivuljo. Za to je najprej pomembno, da znamo izračunati število točk na eliptični krivulji oz. moč krivulje, ki jo označimo z $|E(\mathbb{Z}_p)|$. Že Hassejev izrek nam omeji število točk:

Izrek 4.1 *Za eliptično krivuljo E nad $\mathbb{Z}_p$, kjer je p veliko praštevilo, velja*

$$p + 1 - 2\sqrt{p} \leq |E(\mathbb{Z}_p)| \leq p + 1 + 2\sqrt{p}. \tag{4.10}$$

Ta izrek s pridom izkoristi tudi Schoofov algoritem in njegove izpeljanke, s katerimi lahko natančno in učinkovito izračunamo število točk na eliptični krivulji. Več o Schoofovem algoritmu v Barbič [7].

V kriptografiji nad eliptičnimi krivuljami se želimo izogniti t.i. *supersingularnim* eliptičnim krivuljam. Eliptična krivulja nad $\mathbb{Z}_p$ je supersingularna, če je $|E(\mathbb{Z}_p)| \equiv 1 \bmod p$.

Ker desna stran enačbe eliptične krivulje (glej enačbo (4.2)) predstavlja enačbo tretje stopnje za x, je njena diskriminanta enaka $4a^3 + 27b^2$. Če se

želimo izogniti supersingularni eliptični krivulji, mora veljati $4a^3 + 27b^2 \not\equiv 0 \pmod{p}$.

Pri izbiri eliptične krivulje moramo upoštevati sledeče kriterije:

- $|E(\mathbb{Z}_p)|$ mora imeti velik praštevilski faktor, skoraj tako velik kot $|E(\mathbb{Z}_p)|$,
- $|E(\mathbb{Z}_p)| \neq p$,
- red generatorja P, ne sme deliti $p^k - 1$ za $1 \leq k \leq C$, kjer je zgornja meja C v praksi enaka 20. Na ta način postane napad z MOV redukcijo, ki deluje nad supersingularnimi krivuljami, nepraktičen (opisan je v Menezes, Okamoto in Vanstone [21]).

Na tem mestu se v razloge za zgornje kriterije ne bomo spuščali, dovolj je, da vemo, da moramo biti pri izbiri eliptične krivulje pozorni. Več o razlogih za zgornje kriterije v Yin [32].

4.3.1 Diffie-Helmanov dogovor o ključu nad EC

Za Diffie-Helmanov dogovor o ključu, ki je opisan v podpoglavju 3.2, lahko uporabimo grupo točk eliptične krivulje. Dogovor potem izgleda sledeče [31]:

- Anita generira zasebni ključ n_A in izračuna $Q_A = n_A P$, Bojan prav tako generira zasebni ključ n_B in izračuna $Q_B = n_B P$, pri čemer je P generator varne podgrupe.
- Anita pošlje po potencialno nezaščitenem kanalu točko Q_A, Bojan pa točko Q_B. Če je prenos drag ali počasen, se izplača točko kompresirati.
- Anita izračuna točko $n_A Q_B$, Bojan pa točko $n_B Q_A$. Tako imata na koncu oba skupno točko $n_A n_B P$. Nadaljnjo komunikacijo lahko šifrirata z x koordinato skupne točke, ki ji dodata še bit, da določita ali je y pozitiven ali negativen.

4.3.2 Algoritem za digitalno podpisovanje z EC

Algoritem za digitalno podpisovanje nad eliptičnimi krivuljami ECDSA (angl. *Elliptic Curve Digital Signature Algorithm*) je različica algoritma DSA, ki je predstavljen v prejšnjem poglavju. Namesto nad podgrupo moči q v $\mathbb{Z}_p^*$ deluje nad točkami na eliptični krivulji $E(\mathbb{Z}_p)$.

Tudi pri ECDSA bomo predstavili generiranje ključa, podpisovanje in preverjanje podpisa (Jurišić [13]).

Generiranje ključa

Če Anita želi generirati ključ, mora storiti naslednje:

1. Izbere varno eliptično krivuljo E nad $\mathbb{Z}_p$. Število točk na krivulji $|E(\mathbb{Z}_p)|$ mora biti deljivo z velikim praštevilom n.

2. Izbere točko na krivulji $P \in E(\mathbb{Z}_p)$ reda n.

3. Izbere naključno celo število d na intervalu $[2, n-2]$.

4. Izračuna $Q = dP$.

5. Parametri algoritma so (E, P, n), Anitin javni ključ je Q, njen zasebni ključ pa je d.

Podpisovanje

Če želi Anita podpisati sporočilo m, mora storiti naslednje:

1. Izbere naključno naravno število k na intervalu $[2, n-2]$.

2. Izračuna $kP = (x_1, y_1)$ in $r = x_1 \bmod n$. Če je $r = 0$, se vrne na korak 1. (V primeru, da je $r = 0$, enačba za podpisovanje $s = k^{-1}\{H(m) + dr\} \bmod n$ ne vsebuje zasebnega ključa d.)

3. Izračuna $k^{-1} \bmod n$.

4. Izračuna $s = k^{-1}\{H(m)+dr\} \bmod n$, pri čemer je H zgoščevalna funkcija (SHA-1 ali SHA-2).

5. Če je $s = 0$, se vrne na korak 1. (V primeru, da je $s = 0$, namreč $s^{-1} \bmod n$ ne obstaja, ga pa potrebujemo za preverjanje podpisa.)

6. Podpis sporočila m je par števil (r, s).

Preverjanje podpisa

Če želi Bojan preveriti Anitin podpis (r, s) sporočila m, mora storiti sledeče:

1. Na voljo mora imeti parametre (E, P, n), prav tako mora pridobiti Anitin javni ključ Q. Preveri tudi, da sta r in s števili na intervalu $[1, n-1]$.

2. Izračuna $w = s^{-1} \bmod n$ in $H(m)$.

3. Izračuna $u_1 = H(m)w \bmod n$ in $u_2 = rw \bmod n$.

4. Izračuna $u_1P + u_2Q = (x_0, y_0)$ in $v = x_0 \bmod n$.

5. Podpis sprejme, če in samo če je $v = r$.

DSA	ECDSA
q	n
g	P
x	d
y	Q

Tabela 4.2: Prikazane sorodne oznake pri DSA in ECDSA

4.3.3 Napadi na ECDLP

Na tem mestu bomo predstavili napade na ECDLP s poudarkom na algoritmu veliki korak - mali korak. To podpoglavje je povzeto po Kouchaki Barzi [18].

Trenutno ne poznamo nobenega podeksponentnega algoritma za reševanje diskretnega logaritma nad pravilno izbrano eliptično krivuljo (glej zgoraj). Najboljša napada na ECDLP sta algoritem veliki korak - mali korak in Pollardov ρ algoritem. Pri tem imata oba časovno zahtevnost $O(\sqrt{N})$, kjer je N moč grupe na eliptični krivulji. Je pa Pollardov ρ algoritem v primerjavi z algoritmom veliki korak - mali korak, ki porabi $O(\sqrt{N})$ prostora, prostorsko bolj učinkovit. V nadaljevanju je predstavljen algoritem veliki korak - mali korak za eliptične krivulje.

Algoritem veliki korak - mali korak za eliptično krivuljo

Za eliptično krivuljo E nad končnim obsegom $\mathbb{F}_p$, točki $P, Q \in E$, kjer velja $Q = aP$, iščemo ustrezen a:

- najprej izračunamo $m = \lceil \sqrt{(p + 1 + \sqrt{2p})} \rceil$,
- izračunamo tabelo malih korakov jP, kjer je $1 \leq j < m$,
- izračunamo tabelo velikih korakov $Q - imP$, kjer je $i = 0, 1, \ldots, m - 1$ dokler ne najdemo vrednosti, ki je ekvivalentna že naračunani v tabeli malih korakov,
- a potem lahko izračunamo kot $a = im + j \pmod{\text{red}(P)}$.

Tu spet drži, da je algoritem neodvisen od tega, ali najprej izračunamo velike ali male korake. Za ponazoritev neodvisnosti, katere korake izračunamo najprej, smo tu najprej izračunali male korake, medtem ko smo v podpoglavju 3.2 najprej izračunali velike.

Poglavje 5

Implementacija aplikacije

V tem poglavju bomo predstavili aplikacijo, ki je nastala kot izdelek diplomske naloge. Najprej bomo opisali uporabljeno tehnologijo, potem pa oba dela aplikacije: najprej interaktivno predstavitev računanja na eliptični krivulji nad realnimi števili, nato pa še nad praštevilskimi obsegi.

5.1 Uporabljena tehnologija

5.1.1 HTML in CSS

Za samo prikazovanje aplikacije smo si izbrali HTML in CSS [4]. Cilj naloge je, da bi bila aplikacija dostopna vsem zainteresiranim preko spleta, HTML pa je defacto jezik za pisanje spletnih strani. Omogoča nam oblikovanje spletne strani z množico značk, s katerimi lahko oblikujemo zgradbo HTML dokumenta. Ker sam jezik HTML ne ponuja veliko možnosti za urejanje izgleda dokumenta, se skoraj vedno uporablja skupaj s stili CSS, ki nam omogočajo izboljšavo izgleda.

5.1.2 Javascript

Ko brskalnik naloži spletno stran, se ustvari objektni model dokumenta oz. DOM. To je drevesna struktura, katere elementi so značke HTML doku-

menta. Ker nam Javascript [5] omogoča enostavno dostopanje in spreminjanje teh elementov, smo se za interaktivni del aplikacije odločili uporabljati jezik Javascript. Le-ta je objektni skriptni jezik, ki je bil razvit prav z namenom izdelave spletnih strani.

5.1.3 Knjižnice

- Function Plot [2]: Pri izdelavi interaktivne predstavitve eliptičnih krivulj nad realnimi števili smo se odločili za uporabo knjižnice Function Plot, ki omogoča lažje risanje funkcij. Primer risanja z uporabo te knjižnice je prikazan na sliki 5.1.

```
functionPlot({
  target: '#chart',
  disableZoom: true,
  width: 725,
  height: 500,
  xAxis: {
    label: 'x',
    domain: [-8, 8]
  },
  yAxis: {
    label: 'y',
    domain: [-8, 8]
  },
  grid: true,
  data: [
    { fn: 'sqrt(x*x*x + ' + a + '*x + ' + b + ')', skipTip: true},
    { fn: '-sqrt(x*x*x + ' + a + '*x + ' + b + ')', skipTip: true },
    { fn: k + '*x+' + n, skipTip: true},
    {
      points: [
        P, Q, R, Rminus
      ],
      fnType: 'points',
      graphType: 'scatter'
    },

  ],
  annotations: [
    {x: R[0]},
    {x: P[0]}
  ],
});
```

Slika 5.1: Izris eliptične krivulje s knjižnico Function Plot.

- Chart.js [1]: Pri izdelavi interaktivne predstavitve eliptičnih krivulj

nad končnimi obsegi smo se odločili za uporabo knjižnice Chart.js, saj omogoča hitro in lepo izrisovanje grafov raztrosa, kar poenostavljeno točke na eliptični krivulji tudi so. Primer izrisa točk z omenjeno knjižnico je prikazan na sliki 5.2.

```
var ctx = document.getElementById("chart_ec_finite").getContext('2d');
chart = new Chart(ctx, {
    type: 'scatter',
    data: {
        datasets: [{
            data: points,
            pointBackgroundColor: "#698e8e",
            pointRadius: 5,
            id: "all"
        }]
    },
    options: {
        scales: {
            xAxes: [{
                type: 'linear',
                position: 'bottom'
            }],

        },
        animation: false,
        legend: {
            display: false
        },
        onClick: selectPoint,
    }
});
```

Slika 5.2: Izris točk na eliptični krivulji s knjižnico Chart.js.

5.1.4 Heroku

Heroku [3] je oblačna platforma kot storitev oz. PaaS (angl. *Platform as a Service*). Omogoča nameščanje, upravljanje in skaliranje modernih aplikacij v številnih jezikih kot so Ruby, Java, Node.js, Scala, Closure, Python, PHP in Go. Heroku aplikacijo zapakira v kontejner-okolje, ki ima svoj spomin, operacijski sistem in datotečni sistem. Tako se lahko na istem gostitelju neodvisno izvaja več kontejnerjev. Ker je naša aplikacija statična in Heroku načeloma ni namenjen statičnim stranem, smo morali dodati še vrstico v jeziku PHP, ki uporabniku pri vhodu v aplikacijo postreže vhodno stran.

Aplikacija je tako dostopna na `https://ellipticcurves.herokuapp.com/`.

```
<?php include_once("home.html") ?>
```

Slika 5.3: Dodana vrstica PHP-ja zaradi platforme Heroku.

5.2 Interaktivna predstavitev EC nad $\mathbb{R}$

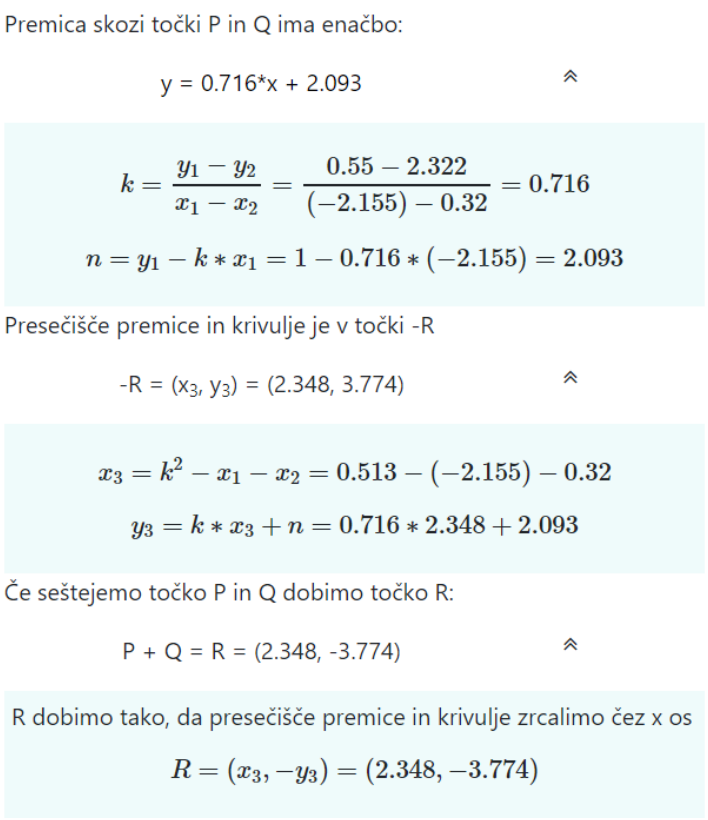
Interaktivna aplikacija uporabniku omogoča izris eliptične krivulje za poljubna parametra a in b, ki ju lahko izbira z drsnikom. Prav tako lahko uporabnik z drsnikom izbira x koordinato točk P in Q na krivulji, v spustnem seznamu pa potem izbere, ali želi, da je izbrana točka $y < 0$ ali $y > 0$. Primer izrisovanja krivulje in seštevanja na eliptični krivulji je prikazan na sliki 4.2, primer drsnikov pa na sliki 5.4.

Slika 5.4: Drsniki za izbiro točk P in Q.

Aplikacija prav tako izračuna enačbo premice skozi točki P in Q, oz. tangente, če je $P = Q$ ter njeno presečišče z eliptično krivuljo, ki je na izrisu označeno z $-R$. Prav tako pojasni izračun točke R. Izračuni in njihova pojasnila se nahajajo na sliki 5.5.

5.3 Interaktivna predstavitev EC nad $\mathbb{Z}_p$

Interaktivna aplikacija uporabniku omogoča izrisovanje točk na eliptični krivulji nad $\mathbb{Z}_p$ za poljubna parametra $a, b \in \mathbb{Z}_p$, ki ju lahko izbira z drsnikom. Prav tako lahko izbira poljuben modulo p, pri čemer aplikacija uporabnika



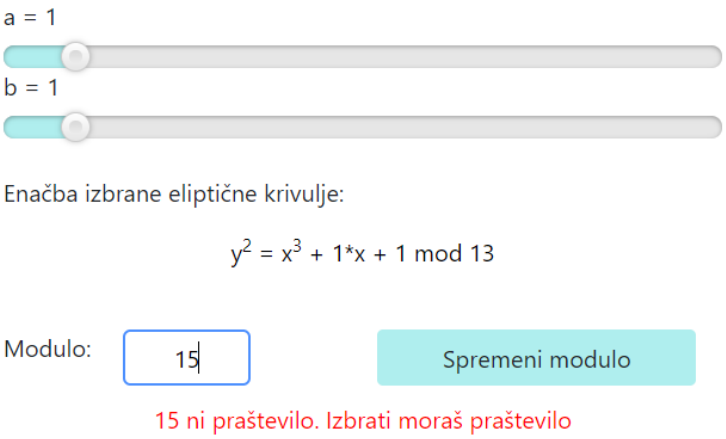
Slika 5.5: Primer izračunov in pojasnil za eliptično krivuljo $y^2 = x^3 - 2x + 6$, kjer je $P = (-2.155, 0.55)$ in je $Q = (0.32, 2.322)$.

opomni, če izbere $p \notin \mathbb{P}$ (prikazano na sliki 5.6). Ker postane izris točk povsem nepregleden pri krivuljah z velikim p, smo omejili, da je največji vnešeni modulo enak 4999.

Aplikacija prav tako prikazuje aritmetiko na eliptični krivulji. Uporabnik lahko s klikom na poljubni točki izbere točki P in Q. Lahko tudi dvakrat klikne na isto točko P. Samodejno se mu izračuna točka R in primerno se na grafu obarvajo točke P, Q in R. Primer seštevanja je prikazan na sliki 4.4.

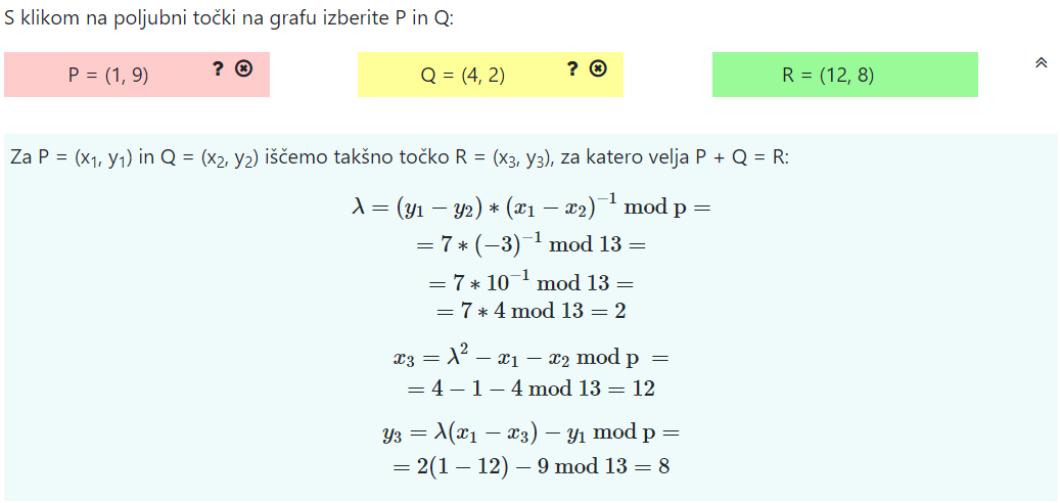
Tako kot pri eliptičnih krivuljah nad realnimi števili, si lahko uporabnik ogleda izračun točke R (glej sliko 5.7).

Če uporabnik klikne na vprašaj pri izpisu točke P in Q, lahko dobi tudi dodatne informacije o točki na krivulji. To so njen inverz, moč podgrupe, ki

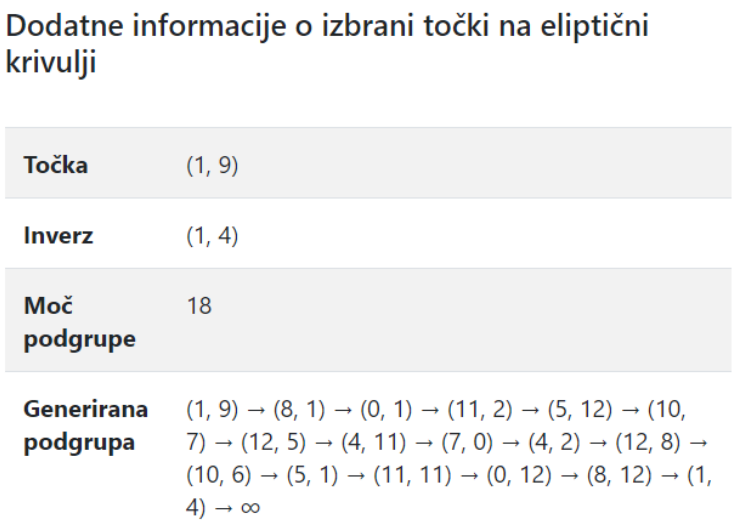


Slika 5.6: Prikazani drsniki za parametra a in b ter okno za izbiranje modula p, ki v tem primeru p ni praštevilo.

jo lahko z dano točko generiramo, in pa izpis vseh točk podgrupe, v istem vrstnem redu, kot jih generiramo. Primer dodatnih informacij o točki je prikazan na sliki 5.8.



Slika 5.7: Posamezni koraki pri računanju točke R na eliptični krivulji $y^2 = x^3 + x + 1$ nad $\mathbb{Z}_{13}$ za točki $P = (1, 9)$ in $Q = (4, 2)$.



Slika 5.8: Dodatne informacije o točki (1, 9) na eliptični krivulji $y^2 = x^3 + x + 1$ nad $\mathbb{Z}_{13}$.

Poglavje 6

Sklepne ugotovitve

V diplomski nalogi smo obravnavali eliptične krivulje ter njihovo uporabo v kriptografiji. Ker varnost kriptografije z eliptičnimi krivuljami temelji na problemu diskretnega logaritma (DLP), saj ne poznamo učinkovitih algoritmov za njegovo reševanje, smo najprej predstavili, kaj je diskretni logaritem. Za začetek smo se osredotočili na DLP v $\mathbb{Z}_p$, njegovo uporabo v kriptografskih protokolih, kot sta Diffie-Helmanov dogovor o ključu ter podpis DSA, prav tako pa smo predstavili dva možna napada na diskretni logaritem, in sicer algoritem veliki korak - mali korak ter Pollardov ρ algoritem s Floydovim algoritmom.

V nadaljevanju smo predstavili eliptične krivulje ter računanje v grupi točk eliptične krivulje. Tu smo se osredotočili na eliptične krivulje nad realnimi števili ter praštevilskimi obsegi $\mathbb{Z}_p$, saj tudi končna aplikacija omogoča vizualizacijo teh dveh tipov eliptičnih krivulj. Predstavili smo tudi analogne kriptografske protokole za delovanje nad eliptičnimi krivuljami, in sicer Diffie-Helmanov dogovor o ključu ter podpis ECDSA.

Končni izdelek diplomske naloge je aplikacija za interaktivno predstavitev eliptičnih krivulj. Sestavljena je iz dveh delov: aplikacije za EC nad realnimi števili ter aplikacije za EC nad $\mathbb{Z}_p$. Prva nam omogoča izris željene eliptične krivulje ter predstavitev grupnega zakona na krivulji. V ta namen nam aplikacija izrisuje vse potrebne tangente, sekante in zrcaljenja čez abscisno

os, prav tako pa nam ponudi vpogled v vse izračune, ki so vodili do končnega rezultata.

Druga aplikacija nam omogoča podobno. V končnih obsegih namreč ne moremo več lepo povezati točk, ker v modularni aritmetiki ni koncepta manjši/večji, zato je v tem primeru eliptična krivulja samo še množica točk v ravnini. Aplikacija nam vse točke, z izjemo točke v neskončnosti, izriše, uporabnik pa lahko potem s klikom izbere poljubni točki, ki ju aplikacija ustrezno obarva in sešteje ter s tem ponazori grupni zakon. Prav tako za poljubno točko izpiše njen inverz, moč generirane podgrupe ter to podgrupo.

Literatura

[1] Chart.js. Dosegljivo: `https://www.chartjs.org/`. [28. 6. 2019].

[2] Function plot. Dosegljivo: `https://mauriciopoppe.github.io/function-plot/`. [28. 6. 2019].

[3] Heroku. Dosegljivo: `https://www.heroku.com/about`. [2. 8. 2019].

[4] Html & css. Dosegljivo: `https://www.w3.org/standards/webdesign/htmlcss`. [28. 6. 2019].

[5] Javascript. Dosegljivo: `https://www.w3schools.com/js/`. [28. 6. 2019].

[6] Kaj je ECC in zakaj ga je smiselno uporabljati? Dosegljivo: `https://www.leaderssl.si/articles/345-kaj-je-ecc-in-zakaj-ga-je-smiselno-uporabljati`. [27. 6. 2019].

[7] J. Barbič. Schoofov algoritem. Diplomska naloga, Fakulteta za matematiko in fiziko, Univerza v Ljubljani, 2000.

[8] M. W. Barsagade in S. Meshram. Overview of history of elliptic curves and its use in cryptography. *International Journal of Scientific & Engineering Research*, **5**(4):467–471, 2014.

[9] J. S. Coron, D. Lefranc in G. Poupard. A new baby-step giant-step algorithm and some applications to cryptanalysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, str. 47–60. Springer, 2005.

[10] W. Diffie in M. Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, **22**(6):644–654, 1976.

[11] S. Friedl. An elementary proof of the group law for elliptic curves. *Groups Complexity Cryptology*, **9**(2):117–123, 2017.

[12] R. Impagliazzo in M. Luby. One-way functions are essential for complexity based cryptography. In *30th Annual Symposium on Foundations of Computer Science*, str. 230–235. IEEE, 1989.

[13] A Jurišić in A. J. Menezes. Elliptic curves and cryptography. *Dr. Dobb's Journal*, str. 26–36, 1997.

[14] A. Jurišić in J. Tonejc. Pametne kartice - 3. del: Napadi in obrambe: velike skrivnosti malih kartic. *Monitor*, **11**(9):44–51, 2001.

[15] A. Jurišić. Računala nove dobe. *Presek*, **30**(4):226–231, 2003.

[16] A. H. Koblitz, N. Koblitz in A. J. Menezes. Elliptic curve cryptography: The serpentine course of a paradigm shift. *Journal of Number theory*, **131**(5):781–814, 2011.

[17] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, **48**(177):203–209, 1987.

[18] B. Kouchaki Barzi. Points of high order on elliptic curves: ECDSA. Magistrsko delo, Univerza Linnaeus, Švedska, 2016.

[19] A. Lenstra in E. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, **14**(4):255–293, 2001.

[20] H. Lenstra. Factoring integers with elliptic curves. *Annals of mathematics*, **126**(3):649–673, 1987.

[21] A. J. Menezes, T. Okamoto in S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on information Theory*, **39**(5):1639–1646, 1993.

[22] A. J. Menezes, P. C. Van Oorschot in S. A. Vanstone. *Handbook of applied cryptography*. CRC press, 1996.

[23] V. Miller. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, str. 417–426. Springer, 1985.

[24] P. Nose. Učinkovita aritmetika na eliptičnih krivuljah nad praštevilskimi obsegi. Diplomska naloga, Fakulteta za računalništvo in informatiko, Fakulteta za matematiko in fiziko, Univerza v Ljubljani, 2008.

[25] National Institute of Standards in Technology. Digital Signature Standard (DSS). Dosegljivo: `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf`. [2. 8. 2016].

[26] P. Paganini. What is a digital signature? Fundamental principles. Dosegljivo: `https://securityaffairs.co/wordpress/5223/digital-id/what-is-a-digital-signature-fundamental-principles.html`. [11. 8. 2016].

[27] D. R. Stinson in M. Paterson. *Cryptography: Theory and Practice*. 2018.

[28] E. Schlegel. Pollardova ρ-metoda. Magistrsko delo, Fakulteta za matematiko in fiziko, Univerza v Ljubljani, 2003.

[29] J. Tonejc. Aritmetika dvojiških končnih obsegov. *Obzornik mat. fiz*, **56**(5):157, 2009.

[30] H. CA Van Tilborg in S. Jajodia. *Encyclopedia of cryptography and security*. Springer Science & Business Media, 2014.

[31] J. Wohlwend. Elliptic curve cryptography: Pre and post quantum. Technical report, MIT, Tech. Rep, 2016.

[32] E. Yin. Curve selection in elliptic curve cryptography. *San Jose State University, Project, Spring*, 2005.