SVEUČILIŠTE U ZAGREBU

PRIRODOSLOVNO–MATEMATIČKI FAKULTET

MATEMATIČKI ODSJEK

Ana Mimica

GENERIRANJE ELIPTIČKIH KRIVULJA METODOM KOMPLEKSNOG MNOŽENJA

Diplomski rad

Voditelj rada:
prof. dr. sc. Andrej Dujella

Zagreb, srpanj 2017.

Ovaj diplomski rad obranjen je dana _____ pred ispitnim povjerenstvom u sastavu:

1. _____, predsjednik
2. _____, član
3. _____, član

Povjerenstvo je rad ocijenilo ocjenom _____.

Potpisi članova povjerenstva:

1. _____
2. _____
3. _____

Mojim roditeljima, bratu Luki i sestri Matei. Za sve. Jer ste bili svugdi u mom svemu. Bez vas ovaj uspjeh ne bi bio potpun.
Veliko hvala prof. dr. sc. Andreju Dujelli na prenesenom znanju i pomoći prilikom izrade ovog diplomskog rada.
Hvala i mojim ljudima na svim zajedničkim sretnim trenutcima.

Sadržaj

Uvod

Kriptografija (kriptós-"skriven" i gráfo-"pisati") je znanstvena disciplina koja se bavi proučavanjem metoda za slanje poruka u izmijenjenom obliku tako da ih samo onaj kome su namijenjene može pročitati. Podrijetlo naziva je grčko, a doslovni prijevod bi bio tajnopis. Razmislimo li na trenutak o značenju doslovnog prijevoda, vrlo brzo ćemo shvatiti važnost uloge kriptografije u današnjem svijetu prepunom strogo čuvanih informacija i podataka. Izrazito je bitno sačuvati sigurnost prilikom izmjene informacija i osigurati da te informacije na odredište stignu bez ikakvih izmjena podataka ili čitanja njihova sadržaja od strane neke treće osobe. U davnim vremenima to se očitavalo u šifriranju pisanih poruka, dok je danas važnost primjene kriptografije vidljiva kroz sve aspekte digitalne komunikacije.

Informacijski sustavi nikada nisu u potpunosti sigurni. Trenutna jačina sigurnosti može lako biti narušena djelovanjem "black hat hackera" koji neprestano usavršavaju svoje vještine i raznim tehnikama uspijevaju probiti šifre, ukrasti podatke, srušiti sustav, itd. Upravo iz tog razloga, potrebna je konstantna nadogradnja sustava u pogledu sigurnosti. Kako bismo onemogućili ili barem otežali dešifriranje tajnih poruka, često u kriptografiji pribjegavamo primjeni matematike i matematičkih problema koji su teško rješivi. Jedna takva primjena se očituje i u korištenju eliptičkih krivulja i teško rješivog problema diskretnog logaritma na eliptičkim krivuljama.

U prvom poglavlju dajemo kratki uvod u kriptografiju i značaj odabira grupe prilikom implementacije, što je moguće sigurnijeg, kriptosustava. Korištena literatura je I.F. Blake, G. Seroussi, N.P. Smart, *Introduction*, Elliptic Curves in Cryptography, Cambridge University Press, Cambridge, 2002, 1-10.

Poglavlje 2 se bavi aritmetikom Weierstrassovih eliptičkih krivulja, a korištena literatura je I.F. Blake, G. Seroussi, N.P. Smart, *Arithmetic on an Elliptic Curve*, Elliptic Curves in Cryptography, Cambridge University Press, Cambridge, 2002, 29-37.

U sljedećem poglavlju obrađujemo osnovne aritmetičke operacije na eliptičkim krivuljama i algoritme za njihovu implementaciju. Također navodimo podjelu točaka prema vrstama koordinata i neke vrste prikaza prirodnog broja kao što su: binarni zapis, SD prikaz i NAF prikaz. Poglavlje 3 se oslanja na literaturu I.F. Blake, G. Seroussi, N.P. Smart, *Efficient Implementation of Elliptic Curves*, Elliptic Curves in Cryptography, Cambridge

University Press, Cambridge, 2002, 57-68 i A.Dujella, *Eliptičke krivulje nad konačnim poljima*, Eliptičke krivulje u kriptografiji, PMF-MO, Zagreb, 2013, 56-59.

Četvrto poglavlje prikazuje primjenu eliptičkih krivulja na faktorizaciji složenih brojeva. Korištena je literatura A.Dujella, *Ostale primjene eliptičkih krivulja*, Eliptičke krivulje u kriptografiji, PMF-MO, Zagreb, 2013, 85-88.

U petom poglavlju analiziramo postupak generiranja eliptičkih krivulja metodom kompleksnog množenja. Prvo smo uveli i definirali sve potrebne pojmove kao što su: j-invarijanta, fundamentalna diskriminanta $-D$, Hilbertov polinom $H_D(x)$, Weberovi polinomi i ostali potrebni pojmovi. Obradili smo Cornacchia algoritam i osvrnuli se na redoslijed konstrukcije krivulja. Konačno slijedi obrada Atkin-Morain metode, ali prije toga imamo kratku analizu sigurnih parametara za eliptičku krivulju. Prilikom svake obrade pojedinog algoritma, procjenjivali smo i njegovu efikasnost odnosno potrebno vrijeme izvođenja. U ovom poglavlju smo koristili literaturu I.F. Blake, G. Seroussi, N.P. Smart, *Generating Curves using Complex Multiplication*, Elliptic Curves in Cryptography, Cambridge University Press, Cambridge, 2002, 149-158; R. Crandall, C. Pomerance, *Elliptic curve arithmetic*, Prime Numbers: A Computational Perspective, Springer Science+Business Media, Inc., New York, 2005, 358-368; A.Dujella, *Kriptosustavi koji koriste eliptičke krivulje*, Eliptičke krivulje u kriptografiji, PMF-MO, Zagreb, 2013, 76-77; D. Hankerson, A. Menezes, S. Vanstone, *Cryptographic Protocols*, Guide to Elliptic Curve Cryptography, Springer-Verlag New York, Inc., New York, 2004, 153-180 i M. Khalgui, O. Mosbahi, A. Valentini, *Securing Embedded Computing Systems through Elliptic Curve Cryptography*, Embedded Computing Systems: Applications, Optimization, and Advanced Desgin, IGI Global, 2013.

Glavni dio ovog rada obrađen je u petom poglavlju dok je u poglavlju 6 prikazan zaključak. Literatura koja je pridonijela stvaranju zaključka je K.Y. Lam, E. Okamoto, C. Xing, Advances in Cryptology - ASIACRYPT' 99, Springer - Verlag Berlin Heidelberg, 1999 i H. Eberle, S. Fung, V. Gupta, N. Gura, S. C. Shantz, D. Stebila, *Speeding up Secure Web Transactions Using Elliptic Curve Cryptography* dostupno na https://www.isoc.org/isoc/conferences/ndss/04/proceedings/Papers/Gupta.pdf (siječanj 2017.).

Poglavlje 1

Uvod u kriptografiju

Osnovni zadatak kriptografije je šifrirati izvornu poruku s ključem tako da primatelj primljene podatke može dešifrirati natrag u izvorni oblik samo ukoliko posjeduje odgovarajući ključ. Primjerice metoda šifriranja simetričnim ključem, koja se i danas primjenjuje u modernoj kriptografiji, koristi jedinstveni ključ što je čini jednostavnom i veoma brzom metodom. Sigurnost je važan aspekt kriptografije, a kod ove metode ona često može biti narušena zbog problema prijenosa ključa. Upravo iz tog razloga se 1976. godine nakon objavljivanja Whitfield Diffievog i Martin Hellmanovog rada pojavilo šifriranje s javnim ključem. Novonastala metoda je uvelike promijenila rad s ključevima pa je opisana kao "najveći, revolucionarni napredak u kriptografiji od renesanse".

Odabir tehnike šifriranja uglavnom ovisi o tome kojem problemu moderne kriptografije dajemo proritet. Najvažniji problemi su:

1. Povjerljivost - poruka koju Alice (pošaljitelj) pošalje Bobu (primatelj) ne smije biti pročitana od neke "treće strane".

2. Autentičnost - Bob zna da je upravo primljenu poruku mogla poslati jedino Alice.

3. Integritet - Bob zna da Aliceina poruka nije bila neovlašteno promjenjena tokom prijenosa.

4. Neodbijanje - Alice poslije ne može tvrditi da ona nije poslala dotičnu poruku.

Tehnike javnog ključa su kod razmatranja problema povjerljivosti obično ograničene u odnosu na simetrično šifrirane, no ostala tri problema mogu biti u korist tehnike javnog ključa kao što je to slučaj s digitalnim potpisima. Kako bi korisniku osigurali nužna svojstva elektroničkog poslovanja (autentičnost, integritet i neodbijanje), digitalni potpisi zahtijevaju korištenje kriptografije javnog ključa.

Sustav koji procesuira bankovna ili poslovna plaćanja će vrlo vjerojatno morati provjeravati ili kreirati tisuće digitalnih potpisa svake sekunde. To nas vodi do zahtjeva za korištenjem shema digitalnih potpisa s javnim ključem radi njihove efikasnosti. I dok su mnoge sheme bazirane na problemu diskretnog logaritma u konačnim Abelovim grupama, postoji rasprava koju vrstu grupe bi bilo najbolje koristiti. Uzimajući u obzir efikasnost, sve više i više postaje popularan izbor grupe točaka na eliptičkoj krivulji nad konačnim poljem. Ovdje primjećujemo povezanost elitpičkih krivulja i kriptografije, a poslije ćemo razmatrati jednu određenu metodu generiranja eliptičkih krivulja i na što točno izbor parametara ima utjecaj.

1.1 Grupe kao osnova kriptografije

Moguće je koristiti različite oznake za grupe ovisno o kontekstu i operaciji. Neki protokoli zahtijevaju upotrebu konačne Abelove grupe G, reda $|G|$, uz pretpostavku da zadovoljava svojstvo iz definicije cikličke grupe. Naše područje interesa će biti aditivna grupa točaka na eliptičkoj krivulji, ali treba spomenuti kako je kod nekih protokola pogodnije pretpostaviti da je riječ o multiplikativnoj grupi s generatorom g i redom $|G|$ koji je prost broj. Ako to nije slučaj, uvijek možemo uzeti podgrupu od G prostog reda bez bitnog gubitka ili smanjenja sigurnosti. U odabranoj grupi G množenje i potenciranje mora biti lako izvedivo, a izračunavanje diskretnog logaritma teško. Također, poželjno je prisustvo mogućnosti generiranja slučajnih elemenata grupe uniformnom distribucijom.

Problem diskretnog logaritma (DLP - discrete logarithm problem) je problem pronalaska najmanjeg prirodnog broja x ukoliko takav broj postoji, a da pritom, za dane elemente h i g iz grupe G, zadovoljava jednadžbu

$$h = g^x.$$

Zbog pretpostavke o prostosti reda grupe G, ovakav diskretni logaritam uvijek postoji. Cilj kriptografije je otežati rješavanje spomenutog problema. Sve kriptografske tehnike imaju jednu zajedničku osobinu: ako postoji brz način rješavanja DLP u grupi G, tada ona nije sigurna za G. Može se reći da je DLP na neki način mjerilo sigurnosti sustava.
Razbijanje pojedinih kriptosustava zasnovanih na eliptičkim krivuljama nad $\mathbb{Z}/n\mathbb{Z}$, gdje je n produkt dva prosta broja, je jednako teško kao faktorizacija od n. Postoje i kriptosustavi zasnovani na eliptičkim krivuljama nad $\mathbb{Z}/n\mathbb{Z}$ koji su na neki način analogni RSA shemama. Takvi sustavi nemaju težinu razbijanja jednaku faktorizaciji i nemaju nikakvu prednost nad RSA u pogledu sigurnosti, ali zato imaju nešto bolje performanse u odnosu na RSA. Budući je faktorizacija velikih prirodnih brojeva dokazano "težak" matematički problem, nas će zanimati kriptosustavi čije je razbijanje ekvivalentno faktorizaciji broja n.

1.2 Značaj odabrane grupe u praksi

Procijenit ćemo posljedice korištenja grupe racionalnih točaka na pogodno odabranoj eliptičkoj krivulji E definiranoj nad $\mathbb{F}_q$, u oznaci $E(\mathbb{F}_q)$, za implementaciju kriptosustava osnovanog na DLP kao suprotnost "konvencionalnom" odabiru multiplikativne grupe $\mathbb{F}_p^*$ konačnog polja. Glavni predmet promatranja je da za dobro odabranu krivulju, najbolja poznata metoda rješavanja DLP na $E(\mathbb{F}_q)$ ima eksponencijalnu složenost u $n = \lceil \log_2 q \rceil$ dok su algoritmi za DLP u $\mathbb{F}_p^*$ subeksponencijalni u $N = \lceil \log_2 p \rceil$.
Preciznije, najbolji poznati opći algoritmi za ECDLP (problem diskretnog logaritma na eliptičkim krivuljama) imaju složenost proporcionalnu

$$C_{EC}(n) = 2^{n/2}.$$

Definirajmo funkciju

$$L_p(v, c) = exp\Big(c(\ln p)^v (\ln \ln p)^{(1-v)}\Big).$$

U slučaju $v = 1$, funkcija L_p je eksponencijalna po $\ln p$ dok je u slučaju $v = 0$ polinomijalna po $\ln p$. Kada je $0 < v < 1$, ponašanje funkcije je strogo između polinomijalnog i eksponencijalnog te se navodi kao sub-eksponencijalno.
Koristeći metodu prosijavanja polja bojeva, diskretne logaritme u $\mathbb{F}_p$ je moguće naći u vremenu proporcionalno $L_p(1/3, c_0)$ pri čemu je $c_0 = (64/9)^{1/3} \approx 1.92$. Složenost izražena u terminima od N, zanemarujući konstantne faktore, iznosi

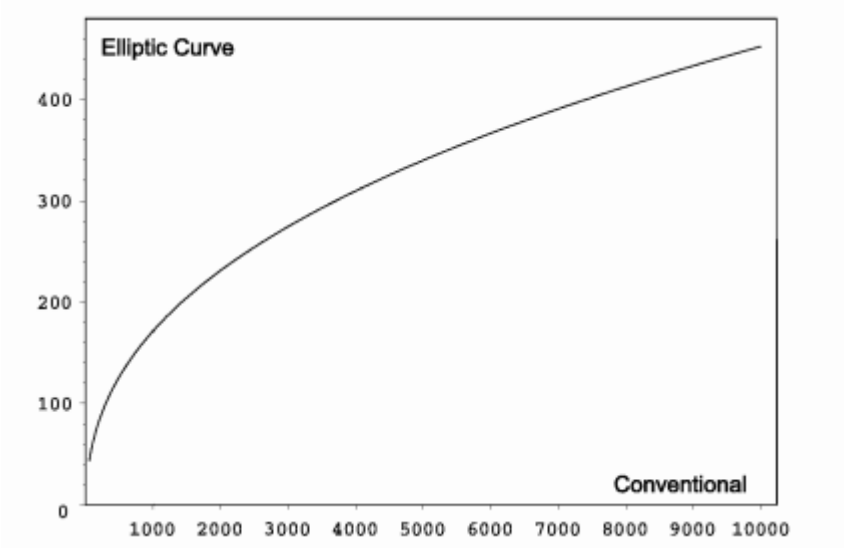
$$C_{CONV}(N) = exp\Big(c_0 N^{1/3} (\ln(N \ln 2))^{2/3}\Big).$$

Indeks CONV znači "konvencijalno". Izjednačimo li C_{EC} i C_{CONV} (opet zanemarujemo konstantne faktore) slijedi da za sličnu razinu sigurnosti moramo imati

$$n = \beta N^{1/3} (\ln(N \ln 2))^{2/3}$$

pri čemu je $\beta = 2c_0/(\ln 2)^{2/3} \approx 4.91$.
Sada parametre n i N interpretiramo kao veličinu ključa za odgovarajuće kriptosustave izraženu u bitovima. Za sličnu kriptografsku snagu veličina ključa u kriptosustavu s eliptičkim krivuljama raste brže nego kubni korijen veličine pripadajućeg konvencionalnog ključa. Odnos ključeva je vidljiv na slici 1.1. Veličinama ključa 1024 bita i 4096 bitova u konvencionalnom kriptosustavu, ekvivalentne su veličine ključa u kriptosustavu s eliptičkim krivuljama od 173 bita i 313 bitova u redoslijedu kako su navedene. Za poštenu usporedbu treba uzeti u obzir i složenost implementacije kriptosustava. Pa tako dok je

implementacija potenciranja u grupama približno iste složenosti u oba slučaja, osnovne operacije na grupama su složenije u slučaju eliptičkih krivulja unatoč istoj veličini polja obiju grupa. Ipak, dijagram sa slike 1.1. nam daje očito objašnjenje sve većeg interesa za primjenom eliptičkih krivulja u kriptografiji kao "jeftinije" alternative konvencionalnim sustavima jer u praksi kraća duljina ključa vodi bržim implementacijama, manjoj potrošnji energije, itd.



Slika 1.1: Odnos veličine ključeva u konvencionalnom kriptosustavu i kriptosustavu s eliptičkim krivuljama za sličnu kriptografsku snagu.

Poglavlje 2

Aritmetika eliptičkih krivulja

Neka je K polje, $\overline{K}$ algebarsko zatvorenje od K, i K^* njegova multiplikativna grupa.
Eliptičku krivulju nad K definiramo kao skup rješenja u projektivnoj ravnini $\mathbb{P}^2(\overline{K})$ homogene Weierstrassove jednadžbe oblika

$$E : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3$$

pri čemu su a_1, a_2, a_3, a_4, a_6 elementi polja K. Za jednadžbu E kažemo da poprima oblik *duge Weierstrassove forme*. Krivulja E mora zadovoljavati svojstvo nesingularnosti. To znači da barem jedna parcijalna derivacija jednadžbe zapisane u obliku $F(X, Y, Z) = 0$ mora biti različita od 0.
Često ćemo koristiti afinu verziju Weierstrassove jednadžbe jer nam je pogodnija u tom obliku koji glasi:

$$E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$$

pri čemu je $a_i \in K, \forall i \in 1, ..., 6$. Definiramo sljedeće konstante:

$$\begin{aligned}
&b_2 = a_1^2 + 4a_2, \qquad b_4 = a_1a_3 + 2a_4, \qquad b_6 = a_3 + 4a_6,\\
&b_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2,\\
&c_4 = b_2^2 - 24b_4, \qquad c_6 = -b_2^3 + 36b_2b_4 - 216b_6.
\end{aligned}$$

Diskriminanta je sada definirana kao

$$\Delta = -b_2^2b_8 - 8b_4^3 - 27b_6^2 + 9b_2b_4b_6.$$

Kada je karakteristika polja različita od 2 i 3, diskriminatna postaje

$$\Delta = (c_4^3 - c_6^2)/1728$$

(i primjetimo da je $1728 = 2^6 3^4$). Sada možemo iskazati i drugi način definiranja nesingularnosti. Krivulja je nesingularna ako i samo ako joj je diskriminanta $\Delta \neq 0$. U slučaju $\Delta \neq 0$, definiramo j-invarijantu krivulje E u oznaci $j(E)$ kao

$$j(E) = c_4^3/\Delta.$$

J-invarijanta je usko povezana s pojmom izomorfizma eliptičkih krivulja. Dvije eliptičke krivulje definirane Weierstrassovim jednandžbama E (s varijablama X i Y) i E' (s varijablama X' i Y') su izomorfne nad K ako i samo ako postoje konstante $r, s, t \in K$ i $u \in K^*$ takve da dopustivom zamjenom varijabli

$$X = u^2X' + r, \qquad Y = u^3Y' + su^2X' + t$$

jednadžbu E transformiramo u E'. Ova transformacija je reverzibilna i njen inverz također definira dopustivu zamjenu varijabli koja E' mijenja u E. Osim svojstva reverzibilnosti, izmomorfizam je relacija ekvivalencije, a uz to stvara bijekciju između skupa racionalnih točaka na E i skupa racionalnih točaka na E'. Slijedi iskaz leme koja govori da j-invarijanta karakterizira klasu ekvivalencije ove relacije nad algebarskim zatvorenjem $\overline{K}$.

Lema 2.0.1. *Dvije izomorfne eliptičke krivulje nad poljem K imaju jednaku j-invarijantu. Vrijedi obrat, dvije krivulje s jednakom j-invarijantom su izomorfne nad* $\overline{K}$.

Sada nakratko pretpostavimo da je karakteristika polja K različita od 2 i 3, te obavimo zamjenu varijabli u dugoj Weierstrassovoj jednadžbi s

$$X = X' - \frac{b_2}{12}, \qquad Y = Y' - \frac{a_1}{2}(X' - \frac{b_2}{12}) - \frac{a_3}{2},$$

a koeficijent b_2 smo ranije definirali kao $b_2 = a_1^2 + 4a_2$. Ovakvom zamjenom se jednadžba u dugoj Weierstrassovoj formi transformira u jednadžbu izomorfne krivulje dane u *kratkoj Weierstrassovoj formi* oblika

$$E : Y^2 = X^3 + aX + b,$$

za neke $a, b \in K$.

Jedna od osnovnih aritmetičkih operacija je zbrajanje koje ćemo prije formalnog iskazivanja lemom, intuitivno opisati geometrijski. Neka su P i Q dvije različite točke na eliptičkoj krivulji E. Kroz njih provučemo pravac koji ih spaja i koji onda krivulju E presijeca u još jednoj dodatnoj točki R. To će se sigurno dogoditi jer se radi o kubnoj krivulji. Točka R će također biti racionalna budući su pravac, krivulja te točke P i Q definirani nad K. Ako nakon toga zrcalimo točku R preko x-osi, dobivamo novu racionalnu točku na E koju možemo označiti $P + Q$. Za zbrajanje dvije iste točke obavlja se sličan postupak uz malu modifikaciju prvog koraka. Naime, kako bismo dodali točku P samoj

sebi, ili žargonskim rječnikom duplicirali točku P, moramo povući tangnetu na krivulju E u točki P. Tangenta presijeca $E(K)$ u još točno jednoj točki, nazovimo je R, jer je E definirana kubnom jednadžbom. Opet zrcalimo točku R preko x-osi i dobivamo točku $[2]P = P + P$. Ako je povučena tangenta u točki vertikalna, onda ona presijeca krivulju u točki beskonačnosti, i tada je $P + P = O$. Točku $(0, 1, 0)$ koja zadovoljava projektivnu jednadžbu $Y^2Z = X^3 + aXZ^2 + bZ^3$ nazivamo točka beskonačnosti.
Spomenute operacije na točkama mogu poslužiti za definiranje pravila aditivne Abelove grupe na $E(\hat{K})$, za proizvoljno polje $K \subseteq \hat{K} \subseteq \overline{K}$, s točkom beskonačnosti $O = 0$. Pravilo možemo jednostavno formulirati i izjavom: "Suma triju točaka krivulje je nula ako i samo ako sve tri točke leže na pravcu." Formalni iskaz definicije zbrajanja točaka eliptičke krivulje je dan u sljedećoj lemi i formule vrijede za polja proizvoljne karakteristike.

Lema 2.0.2. ***(grupovni zakon)*** *Neka je E eliptička krivulja dana jednadžbom*

$$E : Y^2 + a_1XY + a_3Y = X^3 + a_2X^2 + a_4X + a_6$$

i neka su $P_1 = (x_1, y_1)$ i $P_2 = (x_2, y_2)$ točke na krivulji. Tada je

$$-P = (x_1, -y_1 - a_1x_1 - a_3).$$

Definirat ćemo nove koeficijente λ i μ. U slučaju $x_1 \neq x_2$ neka su

$$\lambda = \frac{(y_2 - y_1)}{(x_2 - x_1)}, \qquad \mu = \frac{(y_1x_2 - y_2x_1)}{(x_2 - x_1)},$$

a kada vrijedi $x_1 = x_2$ i $P_2 \neq -P_1$ tada

$$\lambda = \frac{(3x_1^2 + 2a_2x_1 + a_4 - a_1y_1)}{(2y_1 + a_1x_1 + a_3)}, \qquad \mu = \frac{(-x_1^3 + a_4x_1 + 2a_6 - a_3y_1)}{(2y_1 + a_1x_1 + a_3)}.$$

Kooordinate za točku $P_3 = (x_3, y_3) = P_1 + P_2 \neq O$ su dane formulama

$$x_3 = \lambda^2 + a_1\lambda - a_2 - x_1 - x_2,$$
$$y_3 = -(\lambda + a_1)x_3 - \mu - a_3.$$

Osnova kriptografije eliptičkih krivulja je preslikavanje krivulje u samu sebe. Preslikavanje ukratko nazivamo množenje s m, u oznaci $[m]$, za neki prirodan broj m. Množenje točke P s m preslikava točku P u $P+P+\ldots+P$ (m pribrojnika P). Oznaku $[m]$ proširujemo i na $m \leq 0$ definirajući $[0]P = O$ i $[-m]P = -([m]P)$. Sada primjerice množimo točku P na sljedeći način: $[2]P = P + P$, $[3]P = P + P + P$, $[-3]P = -(P + P + P)$, itd ...

2.1 Eliptičke krivulje nad konačnim poljima

Broj racionalnih točaka krivulje nad konačnim poljem $\mathbb{F}_q$ odnosno red grupe $E(\mathbb{F}_q)$ je također konačan i označavat ćemo ga $|E(\mathbb{F}_q)|$. Ako je polje konačno, tada je i red grupe eliptičke krivulje nad tim poljem uvijek konačan, a čine ju sve točke (x, y) koje zadovoljavaju afinu jednadžbu plus točka beskonačnosti. Preciznu ocjenu reda grupe $E(\mathbb{F}_q)$ daje **Hasseov teorem**. Prije njegova iskaza definirat ćemo veličinu $t = q + 1 - |E(\mathbb{F}_q)|$ koju zovemo ***Frobeniusov trag***.

Teorem 2.1.1. ***(Hasse)*** *Frobeniusov trag zadovoljava* $|t| \leq 2\sqrt{q}$.

Za krivulje nad $\mathbb{F}_p$, p prost broj, postoji eliptička krivulja za bilo koji dani red iz intervala $(p + 1 - 2\sqrt{p}, p + 1 + 2\sqrt{p})$. U podintervalu $(p + 1 - \sqrt{p}, p + 1 + \sqrt{p})$ svaki se red pojavljuje s gotovo uniformnom distribucijom. Ta činjenica je osnova koja se krije iza Lenstrinog algoritma za faktorizaciju, takozvanoga ECM-a. Spominjali smo već kako je jačina kriptosustava povezana s faktorizacijom velikih prirodnih brojeva. Postoje dvije klase krivulja koje su se pod određenim okolnostima pokazale kriptografski slabe pa ih se stoga izbjegava koristiti. Riječ je o ***anomalnim*** i ***supersingularnim krivuljama***. Eliptička krivulja $E(\mathbb{F}_q)$ je anomalna ako joj je *Frobeniusov trag* jednak 1 što povlači $|E(\mathbb{F}_q)| = q$. Ova klasa krivulja je slaba ukoliko vrijedi $q = p$ za neki prost broj p. Eliptička krivulja $E(\mathbb{F}_q)$ je supersingularna ako karakteristika p dijeli *Frobeniusov trag* t. Lako se pokaže da je krivulja nad $\mathbb{F}_q$ s karakteristikom p supersingularna ako i samo ako vrijedi jedan od slučajeva:

1. $p = 2$ ili 3 i $j(E) = 0$
2. $p \geq 5$ i $t = 0$.

Nećemo obrađivati sve napade na ove klase krivulja, ali zanimljivo je spomenuti jedan od napada, takozvani MOV (Menezes-Okamoto-Vanstone) napad, koji ima osobitog utjecaja na supersingularne krivulje i čini ih potpuno nepogodnima za kriptografske potrebe. MOV napad reducira ECDLP na DLP u $\mathbb{F}_{q^m}^*$ za neki m. Dok god je m mali, DLP može biti poprilično brzo riješen *index calculus metodom*. Mali m se uvijek može postići za određene vrste krivulja, među kojima su i supersingularne krivulje.

Poglavlje 3

Efikasne implementacije aritmetičkih operacija na eliptičkim krivuljama

Blokovi koji daju temelj za izgradnju kriptosustava zasnovanog na eliptičkim krivuljama nad poljem $\mathbb{F}_q$ su izračunavanja oblika

$$Q = [k]P = P + P + \ldots + P,$$

pri čemu je P točka krivulje, a k proizvoljan cijeli broj, $1 \leq k < |P|$. Za neke kriptografske protokole, točka P je fiksna točka koja je ujedno generator neke podgrupe od $E(\mathbb{F}_q)$ velikog i prostog reda, dok je za druge to proizvoljna točka u podgrupi s netom spomenutim svojstvima. Snaga kripotstava se krije u činjenici da je za danu krivulju, točku P, i točku $[k]P$, teško otkriti cijeli broj k. Izračunavanje k nazivamo ***problem diskretnog logaritma eliptičke krivulje - ECDLP*** (elliptic curve discrete logarithm problem). Kasnije ćemo detaljnije govoriti o ECDLP, a u ovom poglavlju ćemo se usredotočiti na pronalazak i analiziranje efikasnih algoritama za izračunavanje za koje će ECDLP biti "težak" problem.

3.1 Zbrajanje točaka

Promatramo zbrajanje točaka iz polja karakteristike $p > 3$. Analiza će biti obavljena pojedinačno za svaku od tri moguće vrste prikaza točke: preko afinih, projektivnih i miješanih koordinata.

3.1.1 Afine koordinate

Neka je polje $K = \mathbb{F}_q$, $q = p^n$, p prost broj, $p > 3$, n cijeli broj, $n \geq 1$. Jednadžba krivulje u ovom slučaju ima pojednostavljeni oblik - *kratku Weierstrassovu formu*

$$E_{a,b} : y^2 = x^3 + ax + b$$

$a, b \in \mathbb{F}_q$. Sada je diskriminanta krivulje reducirana na $\Delta = -16(4a^3 + 27b^2)$, a j-invarijanta na $j(E) = -1728(4a)^3/\Delta$. U ovom slučaju i izomorfizam klasa krivulje možemo karakterizirati novim načinom. Relacija klasa izmorfizma glasi:

$$E_{a,b} \cong E_{a',b'}$$

ako i samo ako $a' = u^4 a$ i $b' = u^6 b$, za neki $u \in K^*$. Formule iz grupovnog zakona leme 2.0.2 se također pojednostavljuju.
Neka su $P_1 = (x_1, y_1)$ i $P_2 = (x_2, y_2)$ točke u $E(\mathbb{F}_q)$ dane u afinim koordinatama. Pretpostavimo da vrijedi: $P_1, P_2 \neq O$ i $P_1 \neq -P_2$. Negacija točke sada se dobiva samo negacijom y koordinate: $-P_1 = (x_1, -y_1)$.
Kada je $x_1 \neq x_2$ postavimo

$$\lambda = \frac{(y_2 - y_1)}{(x_2 - x_1)},$$

a kada je$x_1 = x_2$ i $y_1 \neq 0$

$$\lambda = \frac{(3x_1^2 + a)}{2y_1}.$$

Suma dviju točaka je nova točka $P_3 = (x_3, y_3) = P_1 + P_2$ čije su koordinate dane formulama

$$x_3 = \lambda^2 - x_1 - x_2$$
$$y_3 = (x_1 - x_2)\lambda - y_1.$$

U slučaju $P_1 \neq P_2$, izračunavanje zahtijeva jednu inverziju i 3 množenja. Trošak ovog zbrajanja ćemo označiti $1I + 3M$ gdje I i M označavaju trošak inverzije i množenja respektivno. Za $P_1 = P_2$, trošak dupliciranja točke iznosi $I + 4M$. Prilikom ocijenjivanja složenosti ovih zbrajanja zanemarivali smo trošak množenja s malom konstantom (npr. $2x_1$) a kvadriranje smatrali regularnim množenjem.

3.1.2 Projektivne koordinate

U slučajevima gdje je inverzija značajno skuplja od množenja, korisno je implementirati *projektivne koordinate*. Projektivna točka (X, Y, Z) na krivulji zadovoljava homogenu Weierstrassovu jednadžbu

$$Y^2Z = X^3 + aXZ^2 + bZ^3,$$

i kada je $Z \neq 0$ odgovara afinoj točki $(X/Z, Y/Z)$. Postoji druga projektivna reprezentacija koordinata koja vodi još učinkovitijoj implementaciji grupovnih operacija. Riječ je o *težinskim projektivnim koordinatama* kod kojih trojka (X, Y, Z) odgovara afinim koordinatama $(X/Z^2, Y/Z^3)$ uz uvjet $Z \neq 0$. To je ekvivalentno korištenju težinske projektivne krivulje čija je jednadžba oblika

$$Y^2 = X^3 + aXZ^4 + bZ^6.$$

Težinske projektivne koordinate su veoma prirodne za eliptičke krivulje i preferirat ćemo ih u odnosu na obične projektivne. U daljnjem tekstu ćemo koristit termin 'projektivne' pri tome misleći na 'težinske projektivne'. Spomenimo i da je konverzija afinih koordinata u projektivne trivijalna, dok cijena obratnog smjera iznosi $1I + 4M$.
Ključ razmatranja korištenja projektivnih koordinata je u tome što njihovo zbrajanje može biti učinjeno samo množenjem bez ikakvih inverzija. Ako je pak konačni rezultat potrebno iz nekog razloga prikazati u afinih koordinatama, samo jedna inverzija mora biti obavljena nakon završetka operacije množenja točaka. Eliminacija inverzija nas je suočila s povećanim brojem množenja pa je pogodnost korištenja projektivnih koordinata strogo određena omjerom $I : M$.

Neka je $P_3 = (X_3, Y_3, Z_3)$ zbroj dviju točaka $P_1 = (X_1, Y_1, Z_1)$ i $P_2 = (X_2, Y_2, Z_2)$. U slučaju kada su $P_1, P_2 \neq O$ i $P_1 \neq \pm P_2$, zbrajanje nas košta $16M$. Detaljniji prikaz svih potrebnih zbrajanja koji nas dovode do ovog zaključka se vidi na slici 3.1. Uvjet $P_1 = \pm P_2$ je ekvivalentan tome da je $\lambda_3 = 0$ na slici 3.1. Štoviše, za $\lambda_3 = 0$ uvjet $P_1 = P_2$ je ekvivalentan $\lambda_6 = 0$. Sada je riječ o dupliranju točke koje košta $10M$, a detaljniji opis troška je vidljiv na slici 3.2. Cijena $10M$ može biti reducirana na $8M$ kada je $a = -3$ tako što se u tom slučaju λ_1 preuredi u oblik $\lambda_1 = 3(X_1 - Z_1^2)(X_1 + Z_1^2)$ koji košta $2M$ umjesto $4M$. Ovo nije moguće napraviti uvijek, već samo onda kad se krivulja $E_{a,b}$ može transformirati u sebi izomorfnu krivulju $E_{a',b'}$ s $a' = -3$. To je pak moguće ako i samo ako $-3/a$ ima četvrti korijen u $\mathbb{F}_q$. To vrijedi za otprilike jednu četvrtinu vrijednosti koeficijenta a kada je $q \equiv 1 \pmod 4$ i za jednu polovinu vrijednosti koeficijenta a kada je $q \equiv 3 \pmod 4$.

$\lambda_1 = X_1 Z_2^2$	$2M$
$\lambda_2 = X_2 Z_1^2$	$2M$
$\lambda_3 = \lambda_1 - \lambda_2$	
$\lambda_4 = Y_1 Z_2^3$	$2M$
$\lambda_5 = Y_2 Z_1^3$	$2M$
$\lambda_6 = \lambda_4 - \lambda_5$	
$\lambda_7 = \lambda_1 + \lambda_2$	
$\lambda_8 = \lambda_4 + \lambda_5$	
$Z_3 = Z_1 Z_2 \lambda_3$	$2M$
$X_3 = \lambda_6^2 - \lambda_7 \lambda_3^2$	$3M$
$\lambda_9 = \lambda_7 \lambda_3^2 - 2X_3$	
$Y_3 = (\lambda_9 \lambda_6 - \lambda_8 \lambda_3^3)/2$	$3M$
	$----$
	$16M$

Slika 3.1: Zbrajanje točaka u projektivnim koordinatama

$\lambda_1 = 3X_1^2 + aZ_1^4$	$4M$
$Z_3 = 2Y_1 Z_1$	$1M$
$\lambda_2 = 4X_1 Y_1^2$	$2M$
$X_3 = \lambda_1^2 - 2\lambda_2$	$1M$
$\lambda_3 = 8Y_1^4$	$1M$
$Y_3 = \lambda_1(\lambda_2 - X_3) - \lambda_3$	$1M$
	$-----$
	$10M$

Slika 3.2: Dupliranje točke u projektivnim koordinatama

3.1.3 Miješane koordinate

Neka su nam opet $P_1 = (X_1, Y_1, Z_1)$ i $P_2 = (X_2, Y_2, Z_2)$ dvije točke koje zbrajamo, a njihova suma je novonastala točka $P_3 = (X_3, Y_3, Z_3)$. Specijalan interesantni slučaj nastaje ukoliko je $Z_1 = 1$. Tada je jedna točka dana u afinim koordinatama, a druga u projektivnim. Nemamo samo jednu vrstu koordinata u operaciji pa ovakvo zbrajanje nazivamo pomiješano zbrajanje. Ovaj slučaj se pojavljuje u nekim algoritmima za množenje točaka, a košta $11M$. U tablici 3.1 je ukratko sumiran odnos cijena pojednih operacija za sve vrste koordinata. Uočimo tako da u projektivnim koordinatama dupliciranje točke (kada je $a = -3$) košta upola manje od općenitog zbrajanja dviju različitih točaka, dok je u afinim koordinatama dupliciranje skuplja operacija.

Operacije	Koordinate		
	afine	mix	projektivne
Općenito zbrajanje	$1I + 3M$	$11M$	$16M$
Dupliranje (proizvoljan a)	$1M + 4M$	n/a	$10M$
Dupliranje ($a = -3$)	$1I + 4M$	n/a	$8M$

Tablica 3.1: Odnos cijena aritmetičkih operacija i vrsta koordinata

3.2 Množenje točke s prirodnim brojem

Množenje točke s prirodnim brojem u eliptičkim krivuljama je poseban slučaj općenitog problema potenciranja u Abelovim grupama pa kao takav uživa korist i dobrobit svih tehnika dostupnih za opći problem. U primjenama eliptičkih krivulja često je potrebno izračunati višekratnik neke točke P, tj. točku kP. Neka je k prirodan broj. Zanima nas, krenemo li od broja 1 i u svakom koraku računamo zbroj dva prethodna rezultata, koji je najmanji broj koraka potreban da dosegnemo k. Efikasnost algoritama se očitava u što manjem broju prolaza kroz petlju i reduciranju "skupih" operacija u algortimu nekim "jeftinijima". Iako se opće metode potenciranja mogu upotrijebiti i za množenje točke s prirodnim brojem, korisno je proučiti neke jedinstvene neobične karakteristike eliptičkih krivulja te ih primijeniti tokom računanja u svrhu postizanja bržih algoritama. Prvo, oduzimanje točaka eliptičkih krivulja virtualno košta jednako kao i zbrajanje. Brzi algoritmi se proširuju tako da uključe *addition-subtraction chains* (riječ je o generalizaciji lanca zbrajanja kako bi uključio i oduzimanje) i *signed representations* (riječ je o prikazivanju broja u novom obliku koji može uključivati negativni predznak - minus). Drugo, moramo uzeti u obzir relativnu kompleksnost zbrajanja točaka i dupliciranja točke u ovisnosti o vrsti koordinata kojom je točka reprezentirana. Treće, za određenu familiju eliptičkih krivulja, dostupni su posebni prečaci u izračunavanju koji značajno smanjuju trošak operacije množenja točke s prirodnim brojem.

3.2.1 Binarna metoda

Ovo je najstarija i najjednostavnija efikasna metoda za množenje točke s prirodnim brojem. Riječ je o algoritmu "kvadriraj i množi" (u multiplikativnoj notaciji) odnosno "dupliciraj i zbrajaj" (u aditivnoj notaciji koju koristimo kod eliptičkih krivulja). Još se naziva i "binarne ljestve" jer se oslanja na binarni zapis broja k. Ako npr. želimo izračunati točku $13P$, broj 13 pretvorimo u binarni zapis što je $(1101)_2$. Postoje dva načina izračunavanja točke $13P$. Prvi, ako zapis čitamo s desna na lijevo bi bio

$$13P = P + 2(2P) + 2\Big(2(2P)\Big).$$

Drugi, ako zapis čitamo s lijeva na desno je

$$13P = 2\Big(2(P + 2P)\Big) + P.$$

Algoritme za računanje $Q = kP$, gdje je $k = (k_{n-1}, \ldots, k_0)_2$, možemo prikazati metodama binarne ljestve s desna na lijevo i binarne ljestve s lijeva na desno. Ulazni podatci za oba algoritma su točka P i broj k u binarnom zapisu. Izlazni podatak je točka $Q = kP$.

Algoritam 3.2.1. **Binarne ljestve s desna na lijevo**

1. Postavimo $Q \leftarrow O$, $R \leftarrow P$.
2. Za $i = 0$ do $n - 1$ radi:
   2.1. Ako je $k_i = 1$, tada postavimo $Q \leftarrow Q + R$.
   2.2. Postavimo $R \leftarrow [2]R$.
3. Postavimo $Q \leftarrow Q + R$.
4. Vratimo Q.

Algoritam 3.2.2. **Binarne ljestve s lijeva na desno**

1. Postavimo $Q \leftarrow P$.
2. Za i od $n - 1$ do 0 radi:
   2.1. Postavimo $Q \leftarrow [2]Q$.
   2.2. Ako je $k_i = 1$, tada postavimo $Q \leftarrow Q + P$.
3. Vrati Q.

Obje varijante binarne metode imaju isti broj operacija: $n - 1$ dupliciranja, a množenja onoliko koliko ima jedinica u binarnom zapisu od k (što je $\leq n$, a u prosječnom slučaju je oko $n/2$). Pretpostavimo da je broj jedinica jednak $n/2$ i zanemarujući uvjete složenosti $O(1)$, prosječan broj potrebnih operacije je $1.5nI + 3nM$ za afine koordinate i $10nM$ za projektivne koordinate. Jedina prednost varijante "s lijeva na desno" je u tome što se u koraku 2.2., točki Q dodaje uvijek ista točka P pa se to može pokušati iskoristiti u implementaciji. Broj operacija za računanje kP za eliptičku krivulju nad poljem $\mathbb{F}_q$ je $O(\ln n \ln^2 q)$.

```
def inverz(b, n):
    #extended euclidean algorithm
    x1 = list(); q1 = list()
    b1 = list(); n1 = list()
    n1 = n; b1 = b
    #n1 = x * b1 + q
    while True:
        if n1 % b1 == 0: break
        x = int(n1 / b1)
        q = n1 % b1
        n1.append(n1); b1.append(b1)
```

```
            x1.append(x); q1.append(q)
            n1 = b1; b1 = q
        if q == 1:
            l = 1; d = 0; br = 0
            k = len(n1) - 1
            while k >= 0:
                if br == 0:
                    d = l * x1[k] + d
                    br += 1
                else:
                    l = l + d * x1[k]
                    br -= 1
                k -= 1
            return l
        else:
            return "Nema rjesenja"


def zbrajanje(x1, y1, x2, y2, a, b, p):
    if x1 == 0 and y1 == 0:
        return {'x': x2, 'y': y2}
    elif x2 == 0 and y2 == 0:
        return {'x': x1, 'y': y1}
    elif x1 == x2 and y1 == y2:
        if 2 * y1 == 0: x3 = 0; y3 = 0
        else:
            if p == 1:
                lambda1 = (3 * x1 * x1 + a) / (2 * y1)
                x3 = (lambda1 * lambda1 - 2 * x1)
                y3 = ((x1 - x3) * lambda1 - y1)
            else:
                y = inverz(2 * y1, p)
                assert y not in {'Nema rjesenja'},("Ne postoji \
potreban inverz mod p pa ne mozemo izracunati zbroj tocaka.")
                lambda1 = ((3 * x1 * x1 + a) * y) % p
                x3 = (lambda1 * lambda1 - x1 - x2) % p
                y3 = ((x1 - x3) * lambda1 - y1) % p
    elif x1 == x2 and y1 == -y2:
        return {'x': 0, 'y' : 0}
```

```
    else:
        if p == 1:
            lambda1 = (y2 - y1)/(x2 - x1)
            x3 = (lambda1 * lambda1 - x1 - x2)
            y3 = ((x1 - x3) * lambda1 - y1)
        else:
            x = inverz( x2 - x1, p)
            assert x not in {'Nema rjesenja'},("Ne postoji \
potreban inverz mod p pa ne mozemo izracunati zbroj tocaka.")
            lambda1 = ((y2 - y1) * x ) % p
            x3 = (lambda1 * lambda1 - x1 - x2) % p
            y3 = ((x1 - x3) * lambda1 - y1) % p
    return {'x' : x3, 'y' : y3}


def binarneLjestveSlijevaNadesno(P, k, a, b, p):
    Q = {'x' : P['x'], 'y' : P['y']}
    n = len(k)
    for i in range(n-1, 0, -1):
        Q = zbrajanje(Q['x'], Q['y'], Q['x'], Q['y'], a, b, p)
        if k[i] == '1':
            Q = zbrajanje(Q['x'], Q['y'], P['x'], P['y'], a, b, p)
    return Q


def binarneLjestveZdesnaNalijevo(P, k, a, b, p):
    Q = {'x' : 0, 'y' : 0};
    R = {'x' : P['x'], 'y': P['y']}
    n = len(k);
    k = k[::-1]
    for i in range(0, n-1):
        if k[i] == '1':
            Q = zbrajanje(Q['x'], Q['y'], R['x'], R['y'], a, b, p)
        R = zbrajanje(R['x'], R['y'], R['x'], R['y'], a, b, p)
    Q = zbrajanje(Q['x'], Q['y'], R['x'], R['y'], a, b, p)
    return Q


if __name__ == '__main__':
```

```
    print(zbrajanje(3, 1944,3, 1944,-58347,3954150, 1))
    P = {'x': 3, 'y': 1944}
    print(binarneLjestveSlijevaNadesno(P, "101110", \
-58347,3954150, 1))
    print(binarneLjestveZdesnaNalijevo(P, "101110", \
-58347,3954150, 1))
    print(zbrajanje(0, 2, 0, 2, 7, 4, 247))
    P = {'x': 0, 'y': 2}
    print(binarneLjestveSlijevaNadesno(P, "100", 7, 4, 247))
    print(binarneLjestveZdesnaNalijevo(P, "100", 7, 4, 247))

""""
    Ispis:
    {'x': 219.0, 'y': 1296.0}
    {'x': 651.0, 'y': 15552.0}
    {'x': 651.0, 'y': 15552.0}
    {'x': 142, 'y': 120}
    {'x': 102, 'y': 156}
    {'x': 102, 'y': 156}
""""
```

Listing 3.1: Python kod za algoritme 3.2.1 i 3.2.2.

3.2.2 SD prikaz

Metoda koja će sada biti opisana je opće poboljšanje binarne metode (preciznije jedna od raznih metoda poboljšanja), a zanima nas upravo ona jer je specifična za eliptičke krviulje. Naime, jedna od specifičnosti grupe točaka na eliptičkoj krivulji je da u njoj oduzimanje ne košta ništa više od zbrajanja. Složenost tih dviju operacija je jednaka pa je tako $-(x, y) = (x, -y)$ za neparnu karakteristiku polja, odnosno vrijedi $-(x, y) = (x, x + y)$ u karakteristici 2. Ova činjenica se može iskoristit za efikasnije množenje ako razmotrimo prikaz broja k u obliku $k = \sum_{j=0}^{l} s_j 2^j$ pri čemu je $s_j \in \{-1, 0, 1\}$. Ovakav zapis broja k nazivamo ***SD prikaz od k*** *(signed digit representation)*. Poboljšanje binarne metode postižemo zamjenom binarnog zapisa sa zapisom u kojem je dozvoljeno pojavljivanje znamenki -1, 0, 1 takozvanim SD prikazom. Imamo $2^{l+1} - 1$ brojeva koji se mogu prikazati sa $l + 1$ znamenkom, a $3l + 1$ mogućih kombinacija pa očito SD prikaz nije jedinstven.
Npr. 3 = (0 1 1) = (1 0 -1). Svojstvo nejedinstvenosti sugerira izbor kojim ćemo postići što veći broj nula u prikazu, a to će pak množenje učiniti efikasnijim.

Najučinkovitija vrsta SD prikaza je **NAF prikaz** (non-adjacent form). Kažemo da je SD prikaz *rijedak (sparse)* ili *nesusjedan (non-adjacent)* ako nema susjednih znamenki različitih od 0, odnosno ako je $s_j s_{j+1} = 0$, $\forall j \geq 0$. NAF ima najmanji broj znamenki različitih od nule među svim SD prikazima prirodnog broja k. Drugim riječima, kažemo da NAF ima najmanju težinu. Najviše za jednu znamenku je dulji od najkraćeg SD prikaza od k, a očekivana prosječna težina prikaza mu je $l/3$ dok je kod binarnog prikaza očekivana težina $l/2$. Ova svojstva NAF-a su precizno iskazana u lemi koja slijedi.

Lema 3.2.1. *Svaki cijeli broj k ima jedinstven NAF prikaz. NAF ima najmanju težinu među svim SD prikazima od k, i duži je za najviše jednu znamenku od najkraćeg SD prikaza broja k.*

Sljedeći algoritam za dani ne-negativni cijeli broj u binarnom zapisu $(k_{l-1}, \ldots, k_0)_2$ računa njegov NAF prikaz $(s_l, \ldots, s_0)_2$.

Algoritam 3.2.3. **Konverzija u NAF**

1. Postavimo $c_0 \leftarrow 0$, $k_l \leftarrow 0$, $k_{l+1} \leftarrow 0$.
2. Za i od 0 do l radi:
   2.1. Postavimo $c_{i+1} \leftarrow \lfloor (k_i + k_{i+1} + c_i)/2 \rfloor$.
   2.2. Postavimo $s_i \leftarrow k_i + c_i - 2c_{i+1}$.
3. Vratimo $(s_l s_{l-1} \cdots s_0)$.

Slijedi Python kod za konverziju binarnog broja u NAF prikaz. Ulazni podatak je binarni broj a izlazni podatak je NAF prikaz $s_l s_{l-1} \cdots s_0$. Npr. za ulazni podatak $b = 10$ što je binarni zapis broja 2, program ispisuje $s = 010$. Navest ćemo još nekoliko primjera: $b = 11$ daje $s = 10\text{-}1$, $b = 100$ daje $s = 0100$, $b = 101$ daje $s = 0101$, $b = 1001$ daje $s = 01001$, $b = 111$ daje $s = 100\text{-}1$, itd.

```
from math import floor

def NAFprikaz(b):
    l = len(b)
    k = list(b)
    k.reverse()
    k.append('0')
    k.append('0')
    s = list()
    c = [0] * (l+2)
```

```
    for i in range(0, l+1):
        c[i+1] = floor((int(k[i]) + int(k[i+1]) + c[i])/2)
        s.append(int(k[i]) + c[i] - 2 * c[i+1])
    s.reverse()
    print(s)
    NAF = ""
    for i in range(l+1): NAF += str(s[i])
    return NAF


if __name__ == '__main__':
    b = input("Unesite b: ")
    print(NAFprikaz(b))
```

Listing 3.2: Python kod za NAF konverziju

Jednom kada broj k imamo zapisan u NAF prikazu, prilagodba binarne metode toj formi broja nije nikakav problem. Jednostavno kada naiđemo na negativnu znamenku u prikazu broja, zbrajanje u algoritmu zamijenimo oduzimanjem. Ulazni podatci za oba algoritma su točka P i broj k u SD zapisu. Izlazni podatak je točka $Q = kP$. Analogan postupak vrijedi za binarnu metodu s lijeva na desno. Uz pretpostavku da je prosječna NAF težina jednaka $n/3$ (pri čemu je n duljina binarnog zapisa od k), izračunavanje binarnih metoda košta $(4/3)n(2M + I)$ za afine koordinate i $(25/3)nM$ za projektivne koordinate.

Algoritam 3.2.4. **Binarne ljestve s desna na lijevo s predznakom**

1. Postavimo $Q \leftarrow O$, $R \leftarrow P$.
2. Za $i = 0$ do $n - 1$:
   2.1. Ako je $k_i = 1$, tada postavimo $Q \leftarrow Q + R$.
   2.2. Ako je $k_i = -1$, tada postavimo $Q \leftarrow Q - R$.
   2.3. Postavimo $R \leftarrow [2]R$.
3. Postavimo $Q \leftarrow Q + R$.
4. Vratimo Q.

Algoritam 3.2.5. **Binarne ljestve s lijeva na desno s predznakom**

1. Postavimo $Q \leftarrow P$.
2. Za $i = n - 1$ do 0 radi:
   2.1. Postavimo $Q \leftarrow [2]Q$.
   2.2. Ako je $k_i = 1$, tada postavimo $Q \leftarrow Q + P$.
   2.3. Ako je $k_i = -1$, tada postavimo $Q \leftarrow Q - P$.
3. Vratimo Q.

Poglavlje 4

Faktorizacija pomoću eliptičkih krivulja

Jedna od primjena eliptičkih krivulja je faktorizacija složenih brojeva. Pronalazak netrivijalnih faktora velikih složenih brojeva se smatra teškim problemom i na njegovoj težini su zasnovani neki od najvažnijih kriptosustava s javnim ključem. Metode faktorizacije dijelimo na dvije vrste: opće i specijalne. Kod prvih metoda očekivani broj operacija ovisi samo o veličini broja n, dok kod drugih ovisi i o svojstvima faktora broja n. Jedna opća metoda je dijeljenje broja n sa svim prostim brojevima $\leq n$. Uz pretpostavku da nam je dostupna tablica svih prostih brojeva $\leq n$, broj potrebnih dijeljenja u najlošijem slučaju je oko $2n/\ln n$ što povlači da je složenost ove metode $O(n \ln n)$. Bez dostupne tablice složenost bi bila još veća. Metoda je očigledno vrlo neefikasna za velike n-ove, ali korisna je u kombinaciji s boljim metodama faktorizacije u svrhu uklanjanja eventualnih malih faktora od n. ***Pollardova $p-1$ metoda*** (1974.godine) i ***ECM*** (elliptic curve method, 1987.godine) spadaju u specijalne metode faktorizacije. Polazište *Pollardove $p-1$ metode* je **Mali Fermatov teorem**.

Teorem 4.0.1. ***(Mali Fermatov teorem)*** *Neka je p prost broj. Ako $p \nmid a$, onda je $a^{p-1} \equiv 1 (mod\ p)$. Za svaki cijeli broj a vrijedi $a^p \equiv a (mod\ p)$.*

Neka je n složen broj koji želimo faktorizirati. Neka je p neki prost faktor od n. Vrijedi $a^m \equiv 1 (\text{mod } p)$ za svaki višekratnik m od $p-1$. Ako nađemo m, onda nam $\text{nzd}(a^m - 1, n)$ daje faktor od n za koji se nadamo da će biti netrivijalan. Nama problem predstavlja pronalazak višekratnika od $p-1$ kada ne znamo čak niti sam p. Kada $p-1$ ima samo male proste faktore, postoji efikasan način pronalaska traženog višekratnika. Neka nam je B prozivoljna granica za veličinu tih prostih faktora. Kažemo da je prirodan broj *B-gladak* ako su mu svi prosti faktori $\leq B$. Pretpostavimo još da su i sve potencije prostih brojeva, koje dijele $p-1$, manje ili jednake B. Uzmemo li da je m najmanji zajednički višekratnik

brojeva $1, 2, \ldots, B$, broj operacija za računanje $a^m \equiv n$ je $O(B \ln B \ln^2 n + \ln^3 n)$. Najgori slučaj se dobiva kada je broj $(p-1)/2$ prost pa tada ova metoda nije ništa bolja od običnog djeljenja. Na primjeru 4.0.2. ćemo pokazati opisani postupak *Pollardove* $p-1$ *metode*.

Primjer 4.0.2. *Neka je* $n = 58493$ *i zanima nas njegov rastav na proste brojeve. Odaberimo* $B = 8$ *i* $a = 2$*. Tada je* $m = 2^3 \cdot 3 \cdot 5 \cdot 7 = 840$*. Tražimo koliko je* a^m *mod n, tj.* 2^{840} *mod* $58493 = 22186$*. Jedan faktor izračunamo preko nzd*$(a^m - 1, n) =$ *nzd*$(22185, 58493) = 29$*. Zaista,* $n = 29 \cdot 2017$*. Pronašli smo traženi rastav na proste faktore.*

Uspjeh metode ovisi direktno o glatkoći broja $p-1$. Postoje razne varijante ove metode koje koriste glatkoću brojeva $p+1$, p^2+p+1 ili p^2-p+1, a najvažnija je *Lenstrina modifikacija* koja koristi eliptičke krivulje. Riječ je o zamjeni grupe $\mathbb{F}_p^*$ reda $p-1$ grupom $E(\mathbb{F}_p)$ čiji red varira unutar intervala $\langle p+1-2\sqrt{p}, p+1+2\sqrt{p}\rangle$. Cilj je pronaći eliptičku krivulju nad $\mathbb{F}_p$ dovoljno glatkog reda. Radit ćemo s eliptičkim krivuljama nad prstenom $\mathbb{Z}_n$ i od početka ćemo znati da je n sigurno složen broj. Pretpostavit ćemo da je $\text{nzd}(n, 6) = 1$ te promatrati eliptičke krivulje oblika

$$E_{a,b} : y^2 = x^3 + ax + b,$$

gdje je $\text{nzd}(4a^3 + 27b^2, n) = 1$. U slučaju da je n prost, na eliptičkoj krivulji postoji samo jedna projektivna točka koja ne odgovara nekoj afinoj točki (točka u beskonačnosti). Inače, kada je n složen, takvih točaka može biti više.

Lenstrin algoritam za faktorizaciju nazivamo skraćeno **ECM** (elliptic curve method). Osnovni koraci algoritma su:

1. Moramo odabrati eliptičku krivulju koja će nam odgovarati. Jedan od načina je slučajan odabir elemenata a, x, $y \in \mathbb{Z}_n$ pa izračunavanje koeficijenta b preko formule $b = (y^2 - x^3 - ax) \bmod n$. Neka je $g = \text{nzd}(4a^3 + 27b^2, n)$. Sada imamo tri slučaja:
   - Ako je $1 < g < n$, pronašli smo netrivijalni faktor od n.
   - Ako je $g = n$, biramo nove a, x, y.
   - Ako je $g = 1$, našli smo eliptičku krivulju $E_{a,b}$ nad $\mathbb{Z}_n$ i točku $P = (x, y)$ na njoj.
2. Uzmimo k prirodan broj, k mora biti najmanji zajednički višekratnik brojeva $1, 2, \ldots, B$. Granicu B odabiremo tako da bude prikladna što bi u praksi značilo da obično uzmemo $B = 10000$, pa je po potrebi povećavamo.
3. Ovo je korak u kojem računamo višekratnike točke P. Računamo $[k]P \in E_{a,b}(\mathbb{Z}_n)$ koristeći već znane formule za zbrajanje točaka:

$$(x_3, y_3) = (\lambda^2 - x_1 - x_2 \bmod n, \lambda(x_1 - x_3) - y_1 \bmod n),$$

pri čemu je $\lambda = (3x_1^2 + a) \cdot (2y_1)^{-1} \bmod n$ ako su točke jednake, odnosno $\lambda = (y_1 - y_2)(x_1 - x_2)^{-1} \bmod n$ ako su točke različite.

4. Moguće je da se tokom računanja u trećem koraku dogodi da neki zbroj točaka ne možemo izračunati jer d nema inverz modulo n pa ne možemo izračunati d^{-1}. Tada računamo $g = \text{nzd}(d, n)$. Ako je $g \neq n$, onda smo našli netrivijalni faktor od n.

5. Ako n nismo uspjeli rastaviti na proste faktore, imamo dvije opcije na raspolaganju: odabir nove eliptičke krivulje ili povećanje granice B.

Opisani postupak ćemo ilustrirati primjerom 4.0.3.

Primjer 4.0.3. *Faktorizirajmo broj* $n = 247$.
Odaberimo $B = 3$. *Slijedi* $k = nzv(1, 2, 3) = 6$. *Odaberimo eliptičku krivulju* $y^2 = x^3 + 7x + 4$ *i njenu točku* $P = (0, 2)$. *Trebamo izračunati* $[6]P = [2](P + [2]P)$. *Prvo računamo* $[2]P$. *Riječ je o zbrajanju istih točaka pa po formulama iz koraka 3 dobivamo da je* $\lambda = 7 \cdot 4^{-1}$ *mod* $247 = 7 \cdot 62$ *mod* $247 = 7$ *mod* 247. *Slijedi* $[2]P = (142, 120)$. *Sada računamo* $[3]P = P + [2]P$. *Pripadni* λ *je* $118 \cdot 142^{-1}$ *mod* $247 = 118 \cdot 207$ *mod* $247 = 87$ *mod* 247, *pa je* $[3]P = (93, 39)$. *I konačno* $[6]P = [2]([3]P)$. *Pripadni* λ *je ovaj put* $25954 \cdot 78^{-1}$, *ali* $nzd(78, 247) = 13$ *pa inverz od* 78 *modulo* 247 *ne postoji. Po uputama iz koraka 4, slijedi da je 13 faktor od 247. I zaista,* $247 = 13 \cdot 19$. *Faktorizacija je uspješno privedena kraju.*

Uspjeh ovog algoritma ovisi o broju k. On bi trebao biti višekratnik od $|E(\mathbb{F}_p)|$, pri čemu je p neki prost faktor od n, jer u tom slučaju prilikom računanja $[k]P$ pripadni nazivnik će biti djeljiv s p pa neće bit invertibilan modulo n. Drugim riječima, u $E(\mathbb{F}_p)$ će vrijediti $[k]P = O$. Ocjena složenosti ovisi o optimalnom odabiru granice B jer je broj operacija potrebnih za pokušaj faktorizacije pomoću jedne krivulje proporcionalan s B. Složenost algortima je $e^{(\sqrt{2}+O(1))\sqrt{\ln p \ln \ln p}}$. U najgorem slučaju kada je $p = O(\sqrt{n})$, složenost postaje $e^{O(\sqrt{\ln n \ln \ln n})}$. Dakle, riječ je o subeksponencijalnom algoritmu.

Složenost očito ne ide u prilog ovoj metodi jer postoje algoritmi puno bolje složenosti kao npr. algoritam sita polja brojeva. Važno svojstvo ove metode se krije u činjenici da složenost ovisi o najmanjem prostom faktoru od n. Iz tog razloga ona nije prikladna za faktorizaciju RSA modula, odnosno brojeva oblika $n = pq$, gdje su p i q bliski prosti brojevi (prosti brojevi približno jednake veličine), ali zato ECM kod faktorizacije "slučajnih" brojeva često daje bolje rezultate od ostalih metoda. To se događa jer "slučajni" brojevi obično imaju neki prost faktor koji je znatno manji od n. ECM nalazi svoju primjenu i kao pomoćna metoda unutar drugih metoda za faktorizaciju s boljom složenosti jer je unutar tih algoritama potrebno faktorizirati neke pomoćne brojeve za koje možemo očekivati da se ponašaju kao "slučajni".

Poglavlje 5

Generiranje eliptičkih krivulja metodom kompleksnog množenja

5.1 Teorija kompleksnog množenja

Prije analiziranja metoda za odabir sigurnih parametara za eliptičke krivulje, prvo ćemo dati kratak uvod u teoriju kompleksnog množenja. Već je spomenuto ranije da su dvije eliptičke krivulje izomorfne nad algebarskim zatvorenjem ako imaju jednake j-invarijante. Prilikom razmatranja j-invarijanti, potrebno je izdvojiti dva specijalna slučaja: $j = 0$ i $j = 1728$.

Za danu kompleksnu j-invarijantu, eliptičku krivulju nad $\mathbb{C}$ s tom j-invarijantom možemo vrlo jednostavno zapisati pridržavajući se sljedećih pravila:

- ako je $j = 0$ tada koristimo jednadžbu

$$E : Y^2 = X^3 - 1$$

- ako je $j = 1728$ tada koristimo jednadžbu

$$E : Y^2 = X^3 - X$$

- inače postavimo $c = j/(j - 1728)$ i koristimo

$$E : Y^2 = X^3 - 3cX + 2c.$$

Ovaj rezultat ćemo ponovo spominjati u lemi o svojstvima eliptičke krivulje nad danim poljem $\mathbb{F}_p$.

Promotrimo na trenutak pojam endomorfizma od E za eliptičku krivulju E definiranu nad poljem kompleksnih brojeva $\mathbb{C}$. ***Endomorfizam*** je homomorfizam eliptičke krivulje u samu sebe određen racionalnim funkcijama. Skup svih endomorfizama označavamo **End(*E*)**. Takav skup prirodno ima strukturu prstena. Nadalje, aritmetička operacija zbrajanja je izvedena iz eliptičkog zbrajanja, dok je množenje sada postalo kompozicija. Dakle, neka je $P \in E$ i neka su ϕ, σ u End(E). Tada je $\phi + \sigma$ endomorfizam na E koji šalje točku P u $\phi(P) + \sigma(P)$, pri čemu je posljednji "+" oznaka eliptičkog zbrajanja. Slično, $\phi \cdot \sigma$ je endomorfizam na E koji šalje P u $\phi(\sigma(P))$.
Ako krivulja E nije supersingularna, tada je njen prsten endomorfizama, End(E), jednak skupu cijelih brojeva $\mathbb{Z}$ ili poredku imaginarnog kvadratnog polja. U slučaju kada je End(E) jednak poredku imaginarnog kvadratnog polja, kažemo da krivulja ima **CM** (complex multiplication - kompleksno množenje). U tom slučaju vrijedi

$$\mathrm{End}(E) \cong \mathbb{Z} + \mathbb{Z}\tau$$

pri čemu je τ kompleksan algebarski broj stupnja 2, takav da je $\tau \in \mathbb{H}$, a pri tome je $\mathbb{H}$ takozvana *Poincareova poluravnina*, gornja polovica kompleksne ravnine (drugim riječima to je pozitivni imaginarni dio). U teoremu 5.1.1 koji slijedi, vidljiva je poveznica između kompleksnog množenja i j-invarijante u slučaju karakteristike nula.

Teorem 5.1.1. *Neka je $\tau \in \mathbb{H}$ i neka je τ kompleksan algebarski broj drugog stupnja. Kada postavimo $E_\tau = \mathbb{C}/(\mathbb{Z} + \mathbb{Z}\tau)$ imamo:*

1. *End(E_τ) je poredak u $\mathbb{Q}(\tau)$ pa stoga E_τ ima kompleksno množenje,*
2. *$j(\tau) = j(E_\tau)$ je algebarski cijeli broj.*

5.2 Fundamentalna diskriminanta i Hilbertov polinom

Negativna diskriminanta D koja odgovara ocijenjenom redu krivulje definirana se kao:

Definicija 5.2.1. *Negativni cijeli broj D je **fundamentalna diskriminanta** ako je neparni dio od D kvadratno slobodan broj i $|D| \equiv 3, 4, 7, 8, 11, 15 (mod\ 16)$.*

Neka je $\mathbb{H}$ već spomenuta Poincareova poluravnina. Slijedi iskaz najbitnijeg teorema cjeline.

Teorem 5.2.2. *Neka je $\tau \in \mathbb{H}$ kompleksni kvadratni broj s diskriminantom $-D$. Dakle $-D$ je diskriminanta primitivne pozitivno definitne kvadratne forme $Q(x, y)$ takve da je τ korijen*

od $Q(x, 1) = 0$. Označimo s h_D broj klasa s diskriminantom $-D$. Tada je $j(\tau)$ algebarski broj stupnja jednakog h_D i njegov minimalni polinom je dan sljedećom jednadžbom

$$H_D(x) = \Pi(x - j(\alpha))$$

gdje α ide po svim kompleksnim brojevima takvim da je $(\alpha, 1)$ nultočka jedne od h_D neekvivalentnih reduciranih formi s diskriminantom $-D$. $H_D(x)$ još nazivamo ***Hilbertov polinom klasa****.*

Ako je $\mathbb{Z}[\tau]$ maksimalan poredak nekog polja K (polja imaginarnih kvadratnih brojeva), tada je

$$H = K(j(\tau))$$

proširenje polja K stupnja h_D. Ustvari, to je maksimalno nerazgranato Abelovo proširenje od K. U ostatku poglavlja ćemo pretpostavljati da je $\mathbb{Z}[\tau]$ maksimalan poredak nekog imaginarnog kvadratnog polja, a $-D$ će nam predstavljati fundamentalnu diskriminantu. Dakle, fundamentalna diskriminanta $-D$ je kongruentna 1 ili 0 modulo 4 i nijedan neparan prost broj ne dijeli D potencijom većom od jedan. Trebamo navesti da je d kvadratno slobodan prirodan broj takav da $\mathbb{Q}(\tau) = \mathbb{Q}(\sqrt{-d})$, odnosno drugim riječima ako je $d \equiv 3 (\mathrm{mod}\ 4)$, tada je $D = d$, inače $D = 4d$.

U nekim algoritmima se zahtijeva računanje Hilbertovog polinoma klasa iz prethodnog teorema 5.2.2. Točnije, morat ćemo s velikom preciznošću izračunavati vrijednosti od $j(\tau)$ za razne $\tau \in \mathbb{H}$. Znamo da $j(\tau)$ možemo efikasno izračunati preko formula

$$h(\Delta) = \frac{\Delta(2\tau)}{\Delta(\tau)}, \qquad j(\tau) = \frac{(256h(\tau) + 1)^3}{h(\tau)},$$

pri čemu je $\Delta(\tau)$ izračunata koristeći

$$\Delta(\tau) = q\Big(1 + \sum_{n\geq 1}(-1)^n\Big(q^{n(3n-1)/2} + q^{n(3n+1)/2}\Big)\Big)^{24}$$

pri čemu je $q = \exp(2\pi\sqrt{-1}\tau)$. Prema tome $H_D(x)$ se može izračunati procjenjući $j(\alpha)$ i računajući produkt

$$H_D(x) = \prod_{\alpha}(x - j(\alpha)).$$

Ovo može biti izračunato u kompleksnoj floating-point aritmetici, iako s visokom preciznošću budući da koeficijenti od $H_D(x)$ moraju biti cijeli brojevi. Produkt prolazi po svim α oblika

$$\alpha = (-b + \sqrt{-D}/(2a))$$

gdje je $b^2 - 4ac = -D$, $|b| \leq a \leq \sqrt{|D|/3}$, $a \leq c$, $\mathrm{nzd}(a, b, c) = 1$ i ako je $|b| = a$ ili $a = c$, tada je $b \geq 0$. Drugim riječima, $ax^2 + bxy + cy^2$ je primitivna, reducirana pozitivno definitna

binarna kvadratna forma diskriminante $-D$.
Od iznimne važnosti je potreba za visokom preciznošću: budući je $\log|j(\alpha)| \approx \pi\sqrt{D}/a$, koeficijenti od $H_D(x)$ mogu biti ogromni. Potrebna preciznost će biti oko

$$10 + \binom{h_D}{\lfloor h_D/2 \rfloor} \frac{\pi\sqrt{D}}{\log 10} \sum_{\alpha} \frac{1}{a},$$

gdje suma ide po svim skupovima vrijednosti od α kao i u gornjem produktu.

Dosadašnje formule su generalizirane i primjenjive na globalnu teoriju. Za promatranje što se događa nad konačnim poljima, potrebno je specificirati detaljnije gornje konstrukcije. Zbog jednostavnosti ćemo pretpostaviti da je riječ o polju $\mathbb{F}_p$, gdje je p velik prost broj i nas zanimaju krivulje definirane upravo nad tim poljem $\mathbb{F}_p$.
Definirajmo sada funkciju

$$\Delta(q) = q\Big(1 + \sum_{n=1}^{\infty} (-1)^n \big(q^{n(3n-1)/2} + q^{n(3n+1)/2}\big)\Big)^{24}$$

koja će nam trebati u algoritmu 5.2 koji za danu diskriminantu D računa broj klasa i Hilbertov polinom klasa. Ulazni podatak algoritma je fundamentalna diskriminanta D. Algoritam vraća željenu kombinaciju: broj klasa h_D, Hilbertov polinom klasa $T \in \mathbb{Z}[X]$ čiji je stupanj h_D, i skup skraćenog oblika (a, b, c) diskriminante D čija je kardinalnost h_D. Algoritam se sastoji od pet osnovnih koraka. U prvom koraku inicijaliziramo potrebne varijable. Drugi korak predstavlja prolaz kroz petlju dok god je ispunjen zadani uvjet i pri tom računamo potrebne koeficijente. U trećem koraku postavljamo optimalne podatke za polinom (τ, f, j). Tijekom četvrtog koraka računa se T, h_D , i skup (a, b, c), dok se peti korak odnosi na kraj algoritma i izlazne podatke algoritma, a to su h_D, zaokruženi realni dio od T, i skup čiji su elementi oblika (a, b, c). Spomenuti koraci su u "receptu na izvedbu" drugačije numerirani zbog potrebe za while i for petljama.

Algoritam 5.2. ***Broj klasa i Hilbertov polinom klasa***

1. Inicijaliziramo $T \leftarrow 1$, $b \leftarrow D \bmod 2$, $r \leftarrow \lfloor \sqrt{|D|/3} \rfloor$, $h \leftarrow 0$, $skup \leftarrow \{\}$.
2. Dok vrijedi $b \leq r$ radi:
   2.1. Neka je $m = (b^2 - D)/4$
   2.2. Za a od 1 do $\lfloor m \rfloor$ radi:
      2.2.1. Ako je $m \bmod a \neq 0$, idemo na korak 2.2.
      2.2.2. Neka $c = m/a$.
      2.2.3. Ako je $b > a$ idemo na korak 2.2.
      2.2.4. Izračunajmo $\tau = (-b + i\sqrt{|D|})/(2a)$.
      2.2.5. Izračunajmo $f = \Delta(e^{4\pi i\tau})/\Delta(e^{2\pi i\tau})$.
      2.2.6. Izračunajmo $j = (256f + 1)^3/f$.
      2.2.7. Sada računamo izlazne podatke algoritma.
         2.2.7.1. Ako je $b = a$ ili $c = a$ ili $b = 0$ izračunajmo:
         $$T = T \cdot (X - j), h = h + 2, skup = skup \cup (a, b, c).$$
         2.2.7.2. Inače izračunajmo:
         $$T = T \cdot (X^2 - 2Re(j)X + |j|^2), h = h + 2, skup = skup \cup (a, \pm b, c).$$
3. Neka je T' zaokruženi realni dio od $T(x)$.
4. Vratimo $(h, T, skup)$.

U ovom algoritmu u koraku 2.2.5. uočavamo ranije spomenuti problem s floating-point preciznošću zbog funkcije Δ koja mora proći ocjenu za kompleksne argumente q. Dovoljna preciznost cijelog algoritma je u osnovi jednaka

$$\frac{n\sqrt{|D|}}{\ln 10} \sum \frac{1}{a}$$

decimalnih znamenki, gdje suma ide po svim jednostavnim reduciranim formama (a, b, c) diskriminante D. Algoritam se također može koristit i na drugi način, tako da se prvo izračuna samo reducirana forma kako bismo procijenili $\sum 1/a$ i odatle traženu preciznost, a zatim krene ispočetka i ovaj put računa Hilbertov polinom klasa. U svakom slučaju veličina od $\sum 1/a$ je uvijek $O(\ln^2 |D|)$.

Pogledajmo neke primjere polinoma koje daje algoritam. Ovdje se T_D odnosi na Hilbertov polinom klasa za diskriminantu D:

$$\begin{aligned}
&T_{-3} = X,\\
&T_{-4} = X - 1728,\\
&T_{-20} = X^2 - 1264000X - 681472000,\\
&T_{-31} = X^3 + 39491307X^2 - 58682638134X + 1566028350940383.
\end{aligned}$$

Stupanj svakog polinoma je jednak njegovom broju klasa. Vrijedi:

$$\begin{aligned}
h_D = 1 \quad & za \quad D = -3, -4, -7, -8, -11, -19, -43, -67, -163;\\
h_D = 2 \quad & za \quad D = -15, -20, -24, -35, -40, -51, -52, -88, -91,\\
& \qquad -115, -123, -148, -187, -232, -235, -267, -403, -427;\\
h_D = 3 \quad & za \quad D = -23, -31, -59, \ldots
\end{aligned}$$

Lista diskriminanti za $h_D = 1, 2$ je ovdje napisana u cijelosti. Trenutno imamo kompletnu listu samo za $h_D \leq 16$ ali nam je makar poznato kako bismo mogli izračunati listu za bilo koju vrijednost h_D. Efikasno određivanje takve liste je vrlo zanimljiv problem izračunljivosti. Postoje i drugi zanimljivi aspekti ovih polinoma. Konstantni koeficijent polinoma je tako uvijek kub nekog cijelog broja. Također, koeficijenti od T_D rastu radikalno brzo kako prolazimo kroz listu diskriminanti. Zbog tako velikih koeficijenata, ovi polinomi su nepogodni za rukovanje pa neki mogu u Atkin-Morainovu pristupu koristiti manje nezgrapnu Weberovu vrstu polinoma.

5.3 Cornacchia algoritam

Korištena metoda je povezana s aritmetikom kompleksnih kvadratnih polja $K = \mathbb{Q}(\sqrt{-D})$. Osnovni ulazni podatak proceduri je fundamentalna diskriminanta $-D$. Metoda koju ćemo opisati je veoma brza ako je broj klasa, h_D, od K mali. Ipak, dio ljudi izražava zabrinutost za korištenje K s malim brojem klasa. Krivulje nastale u tom slučaju mogu biti pogodnije nekim budućim napadima više no što bi to bile krivulje s općenitijim K. Očekuje se da će prosječno broj klasa h_D rasti kao $O(\sqrt{D})$, pa je mali broj klasa u nekom smislu 'poseban'. Dodatno, za polja s malim brojem klasa je dobro poznato da se lakše koriste u raznim algoritmima (i stoga eventualno u budućnosti u problemu diskretnog logaritma).

Želimo konstruirati krivulju nad $\mathbb{F}_p$ s kompleksnim množenjem čiji je red jednak diskriminanti $-D$, što automatski znači da nećemo konstruirati supersingularne eliptičke krivulje. J-invarijanta takve krivulje je sadržana u polju $\mathbb{F}_p$ pa se od diskriminante $-D$ zahtijeva da

Hilbertov polinom $H_D(x)$ ima korijen modulo p.
Kada je prost broj p nerastavljiv, ne postoji krivulja modulo p s kompleksnim množenjem u $\mathbb{Z}[\sqrt{-D}]$. Iz tog razloga trebamo prost broj p koji se rastavlja u kvadratnom proširenju i za koji je Hilbertovo polje klasa trivijalno kada se promatra lokalno na p. Drugim riječima, tražimo glavni ideal u K s gornjim zahtjevom na p. Kažemo da je I glavni ideal ako postoji neki element $r \in I$ takav da je $I = \langle r \rangle$. Za danu diskriminantu D želimo znati koji prosti brojevi p se rastavljaju u K na proste glavne ideale. Prisjetimo se, ideali su jedna vrsta algebarskih struktura čija definicija glasi:

Definicija 5.3.1. *Neka je R prsten. Podskup $I \subseteq R$ je lijevi (odnosno desni) ideal u R ako su ispunjena sljedeća dva uvjeta:*

1. *I je potprsten od R*
2. *Za sve $r \in R$ i $x \in I$ je $rx \in I$, tj. $RI \subseteq I$ (odnosno za sve $r \in R$ i $x \in I$ je $xr \in I$, tj. $IR \subseteq I$).*

Za približno $1/(2h_D)$ prostih brojeva očekujemo da imaju glavni ideal prvog stupnja koji ih dijeli. Takav prost broj je onaj za koji diofantska jednadžba

$$4p = x^2 + Dy^2$$

ima rješenje. Jedna metoda za njeno rješavanje je ***Cornacchia algoritam*** koji u osnovi računa razvoj u verižni razlomak kvadratnog korijena zadanog racionalnog broja.
Uvođenjem supstitucije $x = 2u$ i $y = 2v$ u gornju jednadžbu $4p = x^2 + Dy^2$ dobivamo ekvivalentnu jednadžbu oblika $p = u^2 + dv^2$ koju možemo riješiti primjenom *Cornacchia algoritma* 5.3. Ulazni podatci su: prirodan broj d koji u svojoj faktorizaciji na proste brojeve ne sadrži niti jedan član s potencijom većom od jedan i prost broj p. Izlazni podatak je rješenje jednadžbe $p = u^2 + dv^2$ ukoliko ono postoji.

Algoritam 5.3. ***Cornacchia algoritam***

1. Neka je x_0 rješenje kongruencije $x^2 \equiv -d \pmod{p}$ i pri tome mora vrijediti $p/2 < x_0 < p$.
2. Postavimo: $p \leftarrow q_0 x_0$, $k \leftarrow 0$.
3. Dok vrijedi $x_k^2 < p < x_{k-1}^2$ radi:
   3.1. Izračunajmo $x_k = q_{k+1}x_{k+1} + x_{k+2}$.
   3.2. Inkrementirajmo varijablu k: k++.
4. Postavimo: $u \leftarrow x_k$, $v \leftarrow \sqrt{(p - x_k^2)/d}$.
5. Ako vrijedi $v \in \mathbb{Z}$ vratimo (u, v); inače vratimo "Jednadžba nema rješenja".

Algoritam 5.3 je pisan "receptom na izvedbu", dok je odgovarajući Python kod prikazan u listingu 5.1. U mainu koda imamo nekoliko poziva metode koja provodi Cornacchia algoritam i pripadne izlazne podatke za svaki od tih poziva.

```
from math import floor, sqrt

def Cornacchia(d, p):
    x = list()
    for x0 in range(int(floor(p / 2)) + 1, p):
        if (x0 * x0 + d) % p == 0:
            x = list(); q = list();
            x.append(x0);
            a = int(p / x0); q.append(a)
            b = int(p % x0); x.append(b)
            x[1] = b
            k = 0
            while True:
                a = int(x[k] / x[k+1])
                b = int(x[k] % x[k+1])
                if b == 0: break
                if x[k] * x[k] < p <= x[k-1] * x[k-1]:
                    break
                q.append(a); x.append(b)
```

```
            k+=1

        u = x[k]; v = ((p - x[k] * x[k]) / d)
        if v < 0:
            if k == 0:
                u = p - x[k]
                v = ((p - u * u) / d)
            else:
                u = x[k-1] - x[k]
                v = ((p - u * u) / d)
        if v < 0: x = list(); q = list(); continue
        v = sqrt(v)
        if not v.is_integer():
            x = list(); q = list(); continue
        else: return {'u' : u, 'v': v}

    if len(x) == 0: return "Jednadzba nema rjesenja"

if __name__ == '__main__':
    print(Cornacchia(1,5))      # (u,v) = (2,1)
    print(Cornacchia(4,5))      # (u,v) = (1,1)
    print(Cornacchia(3,7))      # (u,v) = (2,1)
    print(Cornacchia(5,7))      # Jednadzba nema rjesenja
    print(Cornacchia(6,11))     # Jednadzba nema rjesenja
    print(Cornacchia(7,11))     # (u,v) = (2,1)
    print(Cornacchia(11,11))    # Jednadzba nema rjesenja
    print(Cornacchia(1,17))     # (u,v) = (4,1)
    print(Cornacchia(10,19))    # (u,v) = (3,1)
    print(Cornacchia(30,71))    # Jednadzba nema rjesenja
    print(Cornacchia(35,71))    # (u,v) = (6,1)
    print(Cornacchia(114,127)) # Jednadzba nema rjesenja
    print(Cornacchia(63,127))   # (u,v) = (8,1)
    print(Cornacchia(6,103))    # (u,v) = (7,3)
```

Listing 5.1: Implementacija Cornacchia algoritma u programskom jeziku Python

Prilikom primjene *Cornacchia algoritma* možemo ponavljati unos prostog broja p, nadajući se da je upravo taj p naš traženi input, sve dok algoritam ne pronađe par brojeva (x, y) odnosno (u, v). Prema već spomenutoj ocjeni, očekujemo potrebno isprobavanje $1/(2h_D)$

prostih brojeva prije pronalaska onog odgovarajućeg. Za danu trojku (x, y, p), računamo

$$m = p + 1 \pm x.$$

Ovako izračunati brojevi će biti mogući red grupe eliptičkih krivulja nad $\mathbb{F}_p$ koje ćemo pokušati konstruirati. Provjeravamo je li m pogodan tako što provjerimo ima li m velik prost faktor, je li m različit od p i uz to provjerimo da ne postoji mali broj k takav da je $p^k \equiv 1 \pmod{m}$. Kako bismo objasnili zašto m biramo na ovaj način, prisjetimo se da je broj točaka na krivulji jednak $m = p + 1 - t$ gdje je t Frobeniusov trag. Također se prisjetimo da je $t = \alpha + \overline{\alpha}$, pri čemu je α element u K i norma mu je jednaka p. Rješenje od $x^2 + Dy^2 = 4p$ znači da je $\alpha = \pm(x + \sqrt{-D}y)/2$ element čija je norma jednaka p, s tragom jednakim $\pm x$. Redovi $p + 1 \pm x$ su prema tome redovi eliptičke krivulje i njenog kvadratnog twista. Glavna ideja i sažetak ovog razmatranja, sadržani su u sljedećoj lemi.

Lema 5.3.2. *Eliptičke krivulje nad $\mathbb{F}_p$ imaju sljedeća svojstva:*

- *Svaki element polja $\mathbb{F}_p$ je j-invarijanta eliptičke krivulje nad $\mathbb{F}_p$.*
- *Ako je $D > 4$, tada sve eliptičke krivulje s danom j-invarijantom različitom od 0 i 1728 nad poljem $\mathbb{F}_p$ su dane jednadžbom*

  $$Y^2 = X^3 + 3kc^2X + 2kc^3$$

  gdje je $k = j/(1728 - j)$ i c je bilo koji element polja $\mathbb{F}_p$.
- *Pretpostavimo da krivulje E i E' imaju jednake j-invarijante, ali nisu izomorfne nad $\mathbb{F}_p$. Ako je $j \neq 0$ i $j \neq 1728$, kažemo da je E' kvadratno zakretanje (eng. twist) od E i ako je $|E| = p + 1 - t$, tada je $|E'| = p + 1 + t$.*
- *Kada je $j = 0$ ili $j = 1728$, moramo dodatno promatrati i twist s potencijama 3, 4, 6. Međutim, ovaj slučaj ćemo ignorirati jer su takve krivulje na neki način posebne i stoga bi ih trebalo izbjegavati.*

Pretpostavimo da je $j \neq 0$ i $j \neq 1728$. Posebice, ako je E dana jednadžbom

$$Y^2 = X^3 + aX + b,$$

tada E' može biti dana s

$$Y^2 = X^3 + ac^2X + bc^3$$

gdje je c bilo koji element $\mathbb{F}_p$ koji nije kvadratni ostatak u tom istom polju. To znači, ako j-invarijanta eliptičke krivulje nad $\mathbb{F}_p$ reda m može biti konstruirana, tada postoje dva kandidata za krivulju. Provjeru koji od tih dvaju kandidata ima odgovarajući red možemo

napraviti odabirući nasumične točke krivulja i provjeravajući zadovoljavaju li uvjet $[m]P = O$, jer znamo da ovo mora vrijediti za svaku točku P krivulje E reda m.

Problem se dakle može reducirati na izračunavanje koja j-invarijanta može biti j-invarijanta eliptičke krivulje nad $\mathbb{F}_p$ s danim brojem točaka m i kompleksnim množenjem maksimalnog poredka od $\mathbb{Q}(\sqrt{-D})$. Takva odgovarajuća j-invarijanta mora biti korijen modulo p Hilbertovog polinoma $H_D(x)$. Preostaje izračunati $H_D(x)$ metodom koju smo ranije opisali. Konačno, određivanje korijena polinoma $H_D(x)$ nad poljem $\mathbb{F}_p$ se može napraviti nekom od mnogih tehnika za faktorizaciju polinoma nad konačnim poljima. Sada ćemo obraditi dva primjera za opisani postupak. Odabrali smo primjere 5.3.3 i 5.3.4 s malom diskriminantom koji nam mogu poslužiti i kao test za implementaciju ideja u postupku.

Primjer 5.3.3. *Za prvi primjer uzmimo $D = 8$ i potražimo prost broj p takav da jednadžba*

$$4p = x^2 + Dy^2$$

ima rješenje. Slučajnim izborom prostih brojeva p i primjenom Cornacchia algoritma, pronalazimo rješenje kada je $p = 767787129121$. Rješenje je $(u, v) = (850687, 74262)$, odnosno $(x, y) = (1701374, 148524)$ što nas navodi na pokušaj pronalaska eliptičke krivulje nad $\mathbb{F}_p$ s redom grupe jednakim

$$m = 767785427748,$$

što je $6527844 \cdot 117617$ pri čemu je drugi faktor neparan prost broj. Broj klasa od $K = \mathbb{Q}(\sqrt{-8})$ je jedan, pa je Hilbertov polinom $H_8(x)$ ustvari polinom prvog stupnja. Dobivamo da je $H_8(x) = x - 8000$ što očito ima korijen modulo p. Trebamo eliptičku krivulju s j-invarijantom $j_E = 8000 \equiv 767787137121$. Prema izomorfizmu krivulja nad poljem $\mathbb{F}_p$, postoje dvije takve krivulje s j-invarijantom j_E. Dane su jednadžbama:

$$\begin{aligned} E : Y^2 &= X^3 + 767636601121X \\ &\quad + 138379385121 \end{aligned}$$

$$i$$

$$\begin{aligned} E' : Y^2 &= X^3 + 629519057469X \\ &\quad + 139086763239. \end{aligned}$$

Sada još trebamo odrediti koja od ove dvije krivulje ima red grupe jednak m. Izračunali smo $|E(\mathbb{F}_p)| = 767785427748$, a $|E'(\mathbb{F}_p)| = 767788830496$. Dakle E ima red grupe m i to je tražena eliptička krivulja.

Primjer 5.3.4. *U drugom primjeru uzmimo $D = 292$ čiji je broj klasa četiri. Za prost broj $p = 47106401771464858174371611525 3$ jednadžba*

$$x^2 + Dy^2 = 4p$$

ima rješenje i iz tog rješenja zaključujemo da postoji eliptička krivulja reda grupe jednakog

$$m = 47106401771464763072549858280 2.$$

Hilbertov polinom $H_{292}(x)$ je polinom četvrtog stupnja tj. ima četiri korijena modulo p, i jednak je

$$\begin{aligned} x^4 &- 20628770986042830460800 0x^3 \\ &- 93693622511929038759497066112000000x^2 \\ &+ 45521551386379385369629968384000000000x \\ &- 38025946104251240477999064268800000000000000. \end{aligned}$$

Ta četiri korijena su vrijednosti četiri j-invarijante eliptičkih krivulja nad $\mathbb{F}_p$ čiji je red grupe jednak m. Jedan takav korijen je $j = 952981631055855428990768234 35$, iz kojeg izračunamo dvije eliptičke krivulje:

$$\begin{aligned} E : Y^2 = X^3 &+ 46926843642824672578103513427 7X \\ &+ 15582428504728162327278471776 7 \end{aligned}$$

i

$$\begin{aligned} E' : Y^2 = X^3 &+ 35461873957334781312338909332 4X \\ &+ 31425177859305436259087995457 4. \end{aligned}$$

Prva krivulja ima red $2(p + 1) - m$, a druga krivulja ima red jednak m. Preostala tri korijena od $H_D(x) \pmod p$ također daju povod za pronalazak eliptičkih krivulja nad $\mathbb{F}_p$ reda m. One su:

$$\begin{aligned} E'' : Y^2 = X^3 &+ 22603756783533861156919219889 7X \\ &+ 15069171189022574104612813259 8, \end{aligned}$$

$$\begin{aligned} E''' : Y^2 = X^3 &+ 47056900562677103072152855821 1X \\ &+ 73310635572196048952210986 83, \end{aligned}$$

$$\begin{aligned} E'''' : Y^2 = X^3 &+ 30635306510602611080310530807 4X \\ &+ 20423537673735074053540353871 6. \end{aligned}$$

5.4 Weberovi polinomi

Prilikom korištenja gornje metode nailazimo na dva problema.
Prvi problem je što su stvorene krivulje na neki način 'posebne'. Imat će relativno mali broj klasa, pa bi zbog toga mogle biti pogodnije nekim još nepoznatim, ali u budućnosti prisutnim napadima. Neki smatraju da će postupak čiji su koraci prvo odabir slučajnih krivulja, pa tek onda računanje njihova reda grupe preko npr. Schoofova algoritma, vrlo vjerojatno proizvesti krivulje koje su otporne na neke specijalne napade. Ovo uvjerenje ima široku potporu unutar društva iako još uvijek ne postoji niti jedan dokaz o njegovoj točnosti.

Drugi problem se odnosi na potrebu za velikom preciznošću. Kako bismo izračunali j-invarijantu, moramo prije toga izračunati Hilbertov polinom. To zahtijeva računanje koeficijenata koje može zahtijevati jako visoku preciznost. Ovaj problem možemo riješiti računajući minimalni polinom nekog drugog generatora Hilbertovog polja klasa koji ima poznatu algebarsku vezu s $j(\tau)$. Prednost se očituje u mnogo manjim koeficijentima ovog drugog polinoma što nam omogućava korištenje dosta manje preciznosti. U ovom poglavlju ćemo razmotriti moguće rješenje ovog problema. Riječ je o ***Weberovim polinomima*** za koje ćemo pokazati da imaju prednost nad Hilbertovim, ukoliko promatramo veličinu koeficijenata.

Prvo definiramo *Weberove funkcije*, koristeći *Dedekindovu η-funkciju* $\eta(z)$:

$$h(\tau) = \zeta_{48}^{-1}\frac{\eta((\tau+1)/2)}{\eta(\tau)}, \qquad h_1(\tau) = \frac{\eta(\tau/2)}{\eta(\tau)}, \qquad h_2(\tau) = \sqrt{2}\frac{\eta(2\tau)}{\eta(\tau)}$$

$$\gamma_2(\tau) = \frac{h(\tau)^{24}-16}{h(\tau)^8}, \qquad \gamma_3(\tau) = \frac{(h(\tau)^{24}+8)(h_1(\tau)^8 - h_2(\tau)^8)}{h(\tau)^8},$$

gdje je $\zeta_n = e^{2\pi i/n}$. Ove funkcije nisu sve algebarski nezavisne s obzirom da su sve povezane s j preko jednadžbe

$$j = \frac{(h^{24}-16)^3}{h^{24}} = \frac{(h_1^{24}+16)^3}{h_1^{24}} = \frac{(h_2^{24}+16)^3}{h_2^{24}} = \gamma_2^3 = \gamma_3^2 + 1728.$$

Weber funkciju $\mu(\tau)$ naziva klasa invarijantom ako $\mu(\tau)$ leži u Hilbertovom polju klasa od $\mathbb{Q}(\tau)$. Očito je $j(\tau)$ klasa invarijantna. Međutim, korištenjem Weberovih funkcija možemo odrediti još mnogo klasa invarijanta. Ovo daje povod nastanku novih polinoma. Nazivamo ih ***Weberovi polinomi***, obično ih označavamo $W_D(x)$, i za njihovo računanje koristimo sličnu metodu koju smo koristili za izračunavanje $H_D(x)$. Pronalazak korijena ovih novih polinoma će nam omogućiti obnavljanje j-invarijante. Pri tome se nadamo da će polinomi

imati male koeficijente.

Atkin i Morain predlažu sljedeći način za izbor klasa invarijanta u svrhu stvaranja Weberovih polinoma. Atkin i Morainova sugestija glasi: zapamtite da je $-D$ fundamentalna diskriminanta i uz to imamo da je d kvadratno slobodan prirodan broj takav da je $\mathbb{Q}(\sqrt{-D}) = \mathbb{Q}(\sqrt{-d})$. Primjenjujemo redom sljedeće uvjete (uvjet na djeljivost D s 3 ima najviši prioritet).

- Ako je $D \equiv 3 \pmod 6$, koristimo $\mu = \sqrt{-D}\gamma_3(\tau)$.
- Ako je $D \equiv 7 \pmod 8$, koristimo $\mu = h(\tau)/\sqrt{2}$.
- Ako je $D \equiv 3 \pmod 8$, koristimo $\mu = h(\tau)$.
- Ako je $d \equiv \pm 2 \pmod 8$, koristimo $\mu = h_1(\tau)/\sqrt{2}$.
- Ako je $d \equiv 5 \pmod 8$, koristimo $\mu = h(\tau)^4$
- Ako je $d \equiv 1 \pmod 8$, koristimo $\mu = h(\tau)^2/\sqrt{2}$.

Ipak, postoji jedan problematičan slučaj. Jedini problem ovdje je slučaj kada je $D \equiv 3 \pmod 8$ i $D \not\equiv 3 \pmod 6$. U tom slučaju stupanj $W_D(x)$ je $3h_D$ a ne h_D. Ovo bi mogao biti potencijalni problem, stoga možemo jednostavno ignorirati ovakve diskriminante.

Atkin i Morain također daju detaljan opis kako izračunati razne Weberove polinome i kako koristiti druge alternativne klase invarijanti. Slijedi jedan jednostavan primjer kojim lako možemo predočiti prednost koju nam donose Weberovi polinomi. Uočimo koeficijente polinoma navedenih u primjeru:

$$\begin{aligned} H_{71}(x) = {} & x^7 + 313645809715x^6 - 3091990138604570x^5 + 98394038810047812049302x^4 \\ & - 82353426343973077996809138 9x^3 + 513880036645397678032372632944 6x^2 \\ & - 42531947394613960327460515118765 9x \\ & + 73770708676073111335771424100608 1263 \end{aligned}$$

dok je

$$W_{71}(x) = x^7 + x^6 - x^5 - x^4 - x^3 + x^2 + 2x - 1.$$

5.5 Osvrt na redoslijed konstrukcije krivulja

Upravo smo kroz poglavlja 5.2 i 5.3 pokazali dvije opcije konstrukcije eliptičkih krivulja. Prvo smo pokazali kako za danu fundamentalnu diskriminantu $-D$ izabrati p, postići željeni red grupe i tek tada konstruirati eliptičku krivulju s tim redom grupe. Drugi postupak predlaže slučajni odabir prostog broja p, a zatim rješavanje odgovarajuće diofantske jednadžbe pomoću Cornacchia metode. S obzirom da su neki prosti brojevi pogodniji za implementaciju aritmetičkih zahtjeva potrebnih sustavu eliptičkih krivulja, često je daleko bolje prvo izabrati prost broj pa diskriminantu.

Razmatrajući sve mogućnosti ponašanja racionalnog prostog broja p pri dijeljenju u kvadratnom proširenju $K = \mathbb{Q}(\sqrt{-d})$ i njegovog Hilbertovog polja klasa, možemo postići neka poboljšanja. Rezultat razmatranja i poboljšanja ćemo opisati u sljedećoj lemi:

Lema 5.5.1. *Neka je d kvadratno slobodan i neka je p prost broj takav da je moguće naći rješenje jednadžbe*

$$p = x^2 + dy^2.$$

Tada vrijedi:

- *Ako je $p \equiv 3 \pmod 8$, tada je $D \equiv 2, 3$ ili $7 \pmod 8$.*
- *Ako je $p \equiv 5 \pmod 8$, tada je $D \equiv 1 \pmod 2$.*
- *Ako je $p \equiv 7 \pmod 8$, tada je $D \equiv 3, 6$ ili $7 \pmod 8$.*

Posebno moramo imati $\left(\frac{-d}{p}\right) = \left(\frac{-D}{p}\right) = 1$.

5.6 Generiranje i validacija sigurnih parametra za eliptičku krivulju

Osnova sigurnosti svih kriptografskih sustava zasnovanih na eliptičkim krivuljama se krije u težini problema diskretnog logaritma eliptičke krivulje.

Definicija 5.6.1. ***Problem diskretnog logaritma eliptičke krivulje*** *(ECDLP) glasi: za danu eliptičku krivulju E definiranu nad poljem $\mathbb{F}_q$, točku $P \in \mathbb{F}_q$ reda n, i točku $Q \in \langle P \rangle$, tražimo cijeli broj $l \in [0, n-1]$ takav da je $Q = lP$. Cijeli broj l nazivamo diskretni logaritam od Q po bazi P, u oznaci $l = \log_P Q$.*

Parametri eliptičke krivulje koji će se koristiti u kriptografskim sustavima moraju biti pažljivo odabrani u smislu da budu otporni na sve poznate napade na ECDLP. Jedan veoma nevješt algoritam za rješavanje ECDLP se zasniva na iscrpljujućem pretraživanju dok odabrana točka konačno ne da niz točaka $P, 2P, 3P, 4P, \ldots$ među kojima jedna odgovara točki

Q. Približno vrijeme izvođenja algoritma je n koraka u najgorem slučaju, dok u prosjeku to vrijeme iznosi $n/2$.Ipak, ovakvo iscrpljujuće pretraživanje se može prevariti odabirom parametara eliptičke krivulje s dovoljno velikim n kako bi se količina potrebnog računanja pokazala neisplativom (npr. za $n \geq 2^{80}$). Najbolji poznati napad na ECDLP s općenitom svrhom je kombinacija Pohling-Helman algoritma i Pollardovog rho algoritma, koji ima potpuno eksponencijalno vrijeme izvođenja $O(p)$ pri čemu je p najveći prost djelitelj od n. U svrhu odolijevanja ovom napadu, parametri eliptičke krivulje moraju biti odabrani tako da n bude djeljiv dovoljno velikim prostim brojem p kako bi potreban broj koraka p postao neisplativ za izračun (npr. $p > 2^{160}$).

Spomenimo da ne postoji matematički dokaz o nerješivosti problema diskretnog algoritma eliptičke krivulje jer dosad nitko nije dokazao da ne postoji učinkovit algoritam koji bi riješio ECDLP. Ustvari, postojanje takvog dokaza bi bilo veoma iznenađujuće. Npr., nepostojanje polinomijalnog algoritma za ECDLP bi dalo naslutiti da vrijedi $P \neq NP$ te time riješilo jedno od glavnih neriješenih otvorenih pitanja iz područja teorijskog računarstva. Štoviše, ne postoji teoretski dokaz da ECDLP nije lako obradiv. Npr., nije poznato da je ECDLP *NP-težak* problem, i ne vjeruje se da će biti dokazano da ECDLP jest *NP-težak* problem budući je poznato da bi odlučiva verzija ECDLP pripadala *NP* klasi i *coNP* klasi. Naime, problem je *NP-težak* ako postojanje polinomijalnog algoritma za njegovo rješavanje povlači da je $P = NP$. Zbog vjerovanja da je $P \neq NP$, također se vjeruje da za *NP-teške* probleme ne postoji polinomijalni algoritam. U teoriji složenosti *NP* (nedeterminističko polinomijalno) je skup problema odluke rješivih u polinomijalnom vremenu na nedeterminističkom Turingovom stroju. *P* je klasa složenosti koja sadrži probleme odluke za koje postoje algoritmi koji daju odgovor da/ne, a čije vrijeme izvršavanja ovisi polinomijalno o veličini ulaznih podataka. *NP* se može smatrati kao skup problema odluke (odgovor na problem mora biti: da ili ne) čiji se da-odgovori mogu provjeriti u polinomijalnom vremenu. Problemi čiji se ne-odgovori mogu provjeriti u polinomijalnom vremenu pripadaju *coNP* klasi. Još nije poznato vrijedi li $NP = coNP$. Kako god, postojanje *NP-teškog* odlučivog problema koji je i *NP* i *coNP* bi povuklo da vrijedi $NP = coNP$.

Postoje i tzv. ***izomorfni napadi*** na eliptičke krivulje. Ova vrsta napada pokušava reducirati ECDLP na DLP u izomorfnoj grupi za kojeg su poznati algoritmi čije je vrijeme izvršavanja subeksponencijalno ili brže. Ovoj vrsti napada pripadaju napadi pomoću Weilovog i Tateovog sparivanja, napad na anomalne krivulje nad poljima $\mathbb{F}_p$ gdje je p prost broj, i Weilova metodologija spusta. Matematička pozadina izomorfnih napada je poprilično usavršena i detaljna, ali se zato kriptografske implikacije tih napada mogu jednostavno objasniti. Uz to, postoje protumjere kojima se lako može utvrditi je li dana eliptička krivulja otporna na spomenute napade.

Glavni parametri scheme eliptičke krivulje opisuju eliptičku krivulju E definiranu nad konačnim poljem $\mathbb{F}_q$, osnovnu točku $P \in E(\mathbb{F}_q)$, i njen red n. Važno je da za odabrane parametre ECDLP bude otporan na sve poznate napade i da budu ispoštovani implementacijski zahtjevi i ostala potencijalna ograničenja na sigurnost.

Definicija 5.6.2. *Uređena šestorka $D = (q, a, b, P, n, h)$ predstavlja parametre kriptosustava. Redom ćemo navesti svakog od njih:*

1. *red polja q.*
2. *koeficijenti $a, b \in \mathbb{F}_q$ koji definiraju jednadžbu eliptičke krivulje E nad $\mathbb{F}_q$.*
3. *točka $P = (x_P, y_P) \in E(\mathbb{F}_q)$ prostog reda u afinim koordinatama koju definiraju dva elementa polja $\mathbb{F}_q$: x_P i y_P . P nazivamo osnovna točka.*
4. *n je red točke P.*
5. *$h = |E(\mathbb{F}_q)|/n$.*

Definicija 5.6.3. *U grupi eliptičkih krivulja, red točke P je najmanji prirodan broj n takav da je $[n]P = O$. Ako takav broj ne postoji, kažemo da P ima beskonačan red.*

Radi izbjegavanja Pohlig-Hellman napada i Polardovog rho napada na ECDLP, nužno je da $|E(\mathbb{F}_q)|$ bude djeljiv s dovoljno velikim prostim brojem n. Moramo imati najmanje $n > 2^{160}$. Za fiksno polje $\mathbb{F}_q$, maksimalnu otpornost Pohling-Hellman i Pollardovim rho napadima dosežemo ukoliko krivulju E odaberemo takvu da je $|E(\mathbb{F}_q)|$ prost ili skoro prost broj, odnosno da vrijedi $|E(\mathbb{F}_q)| = hn$ gdje je n prost i h mali (npr. h =1, 2, 3 ili 4). Za izbjegavanje napada na anomalne krivulje nad poljima $\mathbb{F}_q$ gdje je p prost broj, trebamo provjeriti je li ispunjeno $|E(\mathbb{F}_q)| \neq q$. Kako bismo pak izbjegli napade pomoću Weilovog i Tateovog sparivanja, moramo osigurati da n ne dijeli $q^k - 1$ (odnosno mora vrijediti $q^k \not\equiv 1 \mod n$) za sve $1 \leq k \leq C$ gdje je C dovoljno velik da se DLP u $\mathbb{F}_q^*$ smatra neukrotivim (za $n > 2^{160}$ dovoljno je $C = 20$). I konačno, u svrhu izbjegavanja Weilovog spusta, korištenje binarnog polja $\mathbb{F}_{2^m}$ možemo razmatrati kao opciju jedino ukoliko je m prost broj.
Mudar način zaštite od napada na specijalne klase krivulja koji će možda biti otkriveni u budućnosti je odabir eliptičke krivulje E nasumce pod uvjetom da je $|E(\mathbb{F}_q)|$ djeljiv s velikim prostim brojem. S obzirom da je vjerojatnost da slučajno odabrana krivulja podliježe poznatim izomorfnim napadima zanemariva, možemo reći da su poznati napadi kod ovakvog odabira krivulje također spriječeni.

Prikazat ćemo dva algoritma koja se bave problemom slučajnog odabira krivulja. Algoritmi su pisani tzv. "receptom na izvedbu". Prvi algoritam generira eliptičke krivulje nad poljem $\mathbb{F}_p$, dok drugi algoritam provodi provjeru je li eliptička krivulja stvarno slučajno

odabrana. Ulazni podaci prvog algoritma su prost broj $p > 3$ i hash funkcija H veličine l bitova. Izlazni podaci su generator S, te koeficijenti $a, b \in \mathbb{F}_p$ koji određuju eliptičku krivulju $E : y^2 = x^3 + ax + b$.

Algoritam 5.6.1. **Generiranje slučajnih eliptičkih krivulja nad poljem $\mathbb{F}_p$ gdje je p prost broj**

1. Postavimo $t \leftarrow \lceil \log_2 p \rceil$, $s \leftarrow \lfloor (t-1)/l \rfloor$, $v \leftarrow t - sl$.
2. Odaberimo proizvoljan string bitova S duljine $g \geq l$ bitova.
3. Izračunajmo $h = H(S)$, i neka je r_0 string bitova duljine v bitova dobiven uzimanjem najdesnijih v bitova od h.
4. Neka je R_0 string bitova dobiven postavljanjem prvog lijevog bita u r_0 na 0.
5. Neka je z cijeli broj čiji binarni prikaz jest S.
6. Za i od 1 do s radi:
   6.1. Neka je s_i duljine g bitova binarni prikaz cijelog broja $(z + i) \bmod 2^g$.
   6.2. Izračunajmo $R_i = H(s_i)$.
7. Neka je $R = R_0 || R_1 || \ldots || R_s$.
8. Neka je r cijeli broj čiji je binarni prikaz R.
9. Ako je $r = 0$ ili $4r + 27 \equiv 0 \pmod{p}$, idemo na korak 2.
10. Odaberimo proizvoljne $a, b \in \mathbb{F}_p$ koji nisu oba 0, takve da vrijedi $r \cdot b^2 \equiv a^3 \pmod{p}$.
11. Vratimo skup (S, a, b).

Osvrnut ćemo se samo na objašnjenje čemu služit parametar r. Pretpostavimo da je $\mathbb{F}_p$ konačno polje karakteristike > 3. Ako su eliptičke krivulje $E_1 : y^2 = x^3 + a_1 x + b_1$ i $E_2 : y^2 = x^3 + a_2 x + b_2$ definirane nad $\mathbb{F}_p$ izomorfne nad tim istim poljem i vrijedi $b_1 \neq 0$ (pa je i $b_2 \neq 0$), onda vrijedi $a_1^3/b_1^2 = a_2^3/b_2^2$. Singularne eliptičke krivulje su točno one koje imaju ili ($a = 0$ i $b = 0$) ili $a^3/b^2 = -27/4$. Ako je $r \in \mathbb{F}_p$ i $r \neq 0$ i $r \neq -27/4$, tada postoje točno dvije izomorfne klase krivulja $E : y^2 = x^3 + ax + b$ s $a^3/b^2 = r$ u $\mathbb{F}_p$. Stoga, u biti postoje samo dva izbora za (a, b) u 10. koraku algoritma. Uvjeti $r \neq 0$ i $r \neq -27/4$ koji su uvedeni u 9. koraku osiguravaju isključivanje singularne eliptičke krivulje. Konačno, ova metoda za generiranje eliptičkih krivulja neće nikada proizvesti eliptičku krivulju čiji su koeficijenti $a = 0$, $b \neq 0$ ili $a \neq 0$, $b = 0$. Međutim, to nas ne treba zabrinjavati

jer takve krivulje čine zanemariv udio među svim eliptičkim krivuljama, pa je stoga vrlo malo vjerojatno da će ih ijedna metoda koja odabire eliptičke krivulje uniformno slučajnim odabirom ikada generirati.

Ulazni podaci algoritma za provjeru su prost broj $p > 3$, hash funkcija H veličine l bitova, generator S duljine $g \geq l$ bitova, i $a, b \in \mathbb{F}_p$ koji određuju eliptičku krivulju $E : y2^{=}x^3 + ax + b$. Izlazni podatak je string oblika "prihvati" ili "odbij" ovisno je li E stvarno generirana pomoću algoritma 5.6.1.

Algoritam 5.6.2. **Provjera je li eliptička krivulja nad $\mathbb{F}_p$ generirana nasumce**

1. Postavimo $t \leftarrow \lceil \log_2 p \rceil$, $s \leftarrow \lfloor (t-1)/l \rfloor$, $v \leftarrow t - sl$.
2. Odaberimo proizvoljan string bitova S duljine $g \geq l$ bitova.
3. Izračunajmo $h = H(S)$, i neka je r_0 string bitova duljine v bitova dobiven uzimanjem najdesnijih v bitova od h.
4. Neka je R_0 string bitova dobiven postavljanjem prvog lijevog bita u r_0 na 0.
5. Neka je z cijeli broj čiji binarni prikaz jest S.
6. Za i od 1 do s radi:
   6.1. Neka je s_i duljine g bitova binarni prikaz cijelog broja $(z + i) \bmod 2^g$.
   6.2. Izračunajmo $R_i = H(s_i)$.
7. Neka je $R = R_0 || R_1 || \ldots || R_s$.
8. Neka je r cijeli broj čiji je binarni prikaz R.
9. Ako vrijedi $r \cdot b^2 \equiv a^3 \pmod{p}$ vrati ("prihvati"), inače vrati ("odbij").

Jednom kada imamo odabranu eliptičku krivulju možemo generirati i ostale parametre kriptosustava. Ulazni podaci su red polja q, stupanj sigurnosti L koji zadovoljava nejednadžbe $160 \leq L \leq \log_2 q$ i $2^L \geq 4\sqrt{q}$. Izlazni podaci su parametri $D = (q, a, b, P, n, h)$.

Algoritam 5.6.3. **Generiranje parametara kriptosustava**

1. Odaberemo $a, b \in \mathbb{F}_p$ slučajnim odabirom koristeći algoritam 5.6.1. Neka je S izlazni podatak algoritma 5.6.1. Neka je $E : y^2 = x^3 + ax + b$.
2. Izračunamo $N = |E(\mathbb{F}_p)|$.

3. Provjerimo je li N djeljiv s velikim prostim brojem n koji zadovoljava $n > 2^L$. Ako nije, vratimo se na prvi korak.

4. Provjerimo da n ne djeli $q^k - 1$ za sve $1 \leq k \leq 20$. Ako ovaj uvjet nije ispunjen, vratimo se na prvi korak.

5. Provjerimo da vrijedi $n \neq q$. U protivnom, vratimo se na prvi korak.

6. Postavimo $h \leftarrow N/n$.

7. Odaberimo proizvoljnu točku $P \in E(\mathbb{F}_p)$ i postavimo $P = hP'$. Ponavljamo dok god je $P \neq \infty$.

8. Vratimo skup parametara (q, a, b, P, n, h).

Alternativa korištenju slučajno odabranih krivulja je odabir krivulja korištenjem CM metode. Algoritam 5.6.3. se može lako modificirati da se primjenjuje i na tu metodu.

5.7 Atkin-Morain metoda

Bitan dio generiranja parametara kriptosustava je određivanje broja točaka na eliptičkoj krivulji. Za polja čija je veličina od interesa kriptografiji, naivni algoritmi, koji broj točaka određuju tako da za svaki $x \in \mathbb{F}_q$ pronalaze broj rješenja $y \in \mathbb{F}_q$ Weierstrassove jednadžbe, su očito neisplativi. U praksi postoje tri metode kojima odabiremo eliptičku krivulju poznatog reda, a mi ćemo ovdje obraditi jednu takvu metodu. Riječ je o **CM metodi** koja prvo odabire red N koji zadovoljava sigurnosne uvjete, a zatim konstruira eliptičku krivulju tog reda. Za eliptičke krivulje nad poljima $\mathbb{F}_p$ gdje je p prost broj, CM metoda se naziva i **Atkin-Morain metoda**. Za eliptičke krivulje nad poljima $\mathbb{F}_{2^m}$, CM metoda se naziva još i Lay-Zimmer metoda. CM metoda je vrlo učinkovita ako odaberemo polje reda q i eliptičku krivulju reda $N = q + 1 - t$ tako da polje $\mathbb{Q}(\sqrt{t^2 - 4q})$ ima mali broj klasa. Kriptografski pogodne krivulje nad poljima veličine 160 bitova mogu biti generirane u jednoj minuti. Za konačna polja koja mi ovdje promatramo, CM metoda je mnogo brža od svih najboljih poznatih algoritama za računanje točaka na slučajno odabranoj eliptičkoj krivulji. Budući nije poznato da je ECDLP lakše rješiv za eliptičke krivulje koje imaju mali broj klasa, eliptičke krivulje generirane CM metodom nude jednak stupanj sigurnosti kao i one generirane slučajnim odabirom.

CM metoda počinje prostim brojem p i potom odabire najmanju diskriminantu D zajedno s cijelim brojem u koji zadovoljava diofantsku jednadžbu $p = u^2 + dv^2$. Slijedi provjera je li $p + 1 - u$ i/ili $p + 1 + u$ pogodan. Ako nijedan od njih nije pogodan, proces se

ponavlja. Inače, konstruiramo Hilbertov polinom i nađemo mu korijene. Korijen Hilbertovog polinoma je tražena j-invarijanta. Sada konstruiramo eliptičku krivulju i njen twist. Točno jedna od tih dviju krivulja ima tražen pogodan red grupe. Jedan od načina otkrivanja koja krivulja ima traženi red je da odabiremo slučajne točke P na svakoj od krivulja sve dok se ne dogodi da za neku točku na jednoj od krivulja vrijedi $mP \neq O$. U tom slučaju je druga krivulja ona koju mi tražimo.

Sada ćemo prikazati algoritam za generiranje eliptičkih krivulja i njihova reda CM metodom. Kod ovog algoritma imamo dvije mogućnosti, ili pretpostavljamo da je lista diskriminanti konačna ili se moramo složiti da pustimo algoritam da radi neko određeno vrijeme. Za potonji prikaz implementacije algoritma "receptom na izvedbu", odlučujemo se za prvu opciju. Neka je lista fundamentalnih diskriminanti $\{D_j < 0 : j = 1, 2, 3, \ldots, n\}$ uzlazno sortirana prema broju klasa h_D , a u slučaju jednakog broja klasa prema rastućim apsolutnim vrijednostima diskriminante $|D|$. Ulazni podatak algoritma je prost broj $p > 3$. Izlazni podaci su red grupe eliptičkih krivulja ili parametri krivulje povezane s pojedinim D_j . Možemo modificirati što će nam algoritam točno vratit ovisno o tome što želimo saznati.

Algoritam 5.7.1. **CM metoda za generiranje krivulja i njihovog reda**

1. Nasumce odaberimo kvadratni neostatak $g \pmod p$.
2. Ako je $p \equiv 1 \pmod 3$ i $g^{(p-1)/3} \equiv 1 \pmod p$ idemo na korak 1.
3. U slučaju kada je $D = -3$, g ne smije biti kub modulo p. U protivnom, idemo na korak 1.
4. Za j od 0 do n radi:
   - 4.1. Neka je $D = D_j$.
   - 4.2. Ako vrijedi $(\frac{D}{p}) \neq 1$ idemo na korak 1.
   - 4.3. Pokušamo naći reprezentaciju za $4p = u^2 + |D|v^2$ preko Cornacchia algoritma. Ako jednadžba nema rješenja, idemo na korak 1.
   - 4.4. Tražimo mogući red grupe eliptičkih krivulja:
     - 4.4.1. Ako je $D = -4$, tada postavimo $red \leftarrow \{p + 1 \pm u, p + 1 \pm 2v\}$. Imamo četiri mogućnosti za red grupe eliptičkih krivulja.
     - 4.4.2. Ako je $D = -3$, tada postavimo $red \leftarrow \{p + 1 \pm u, p + 1 \pm (u \pm 3v)/2\}$. Imamo šest mogućnosti za red grupe eliptičkih krivulja.
     - 4.4.3. Ako je $D < -4$, tada postavimo $red \leftarrow p + 1 \pm u$. Imamo dvije mogućnosti za red grupe eliptičkih krivulja.

4.5. Tražimo parametre krivulje:

4.5.1. Ako je $D = -4$, postavimo *parametri* $\leftarrow \{(-g^k \bmod p, 0) : \text{za } k = 0, 1, 2, 3\}$.

4.5.2. Ako je $D = -3$, postavimo *parametri* $\leftarrow \{(0, -g^k \bmod p) : k = 0, 1, 2, 3, 4, 5\}$.

4.5.3. Ako je $D < -4$, izračunajmo Hilbertovu klasu polinoma $T \in \mathbb{Z}[X]$ preko algoritma za računanje Hilbertove klase polinoma. Neka je $S = T \bmod p$. Sada možemo dobiti korijen $j \in \mathbb{F}_p$ od S. Izračunajmo $c = j(j - 1728)^{-1} \bmod p$. Izračunajmo $r = -3c \bmod p$. Izračunajmo $s = 2c \bmod p$. Postavimo *parametri* $\leftarrow \{(r, s), (rg^2 \bmod p, sg^3 \bmod p)\}$.

4.6. Vratimo (*red*, *parametri*).

Atkin-Morain metoda propisuje da je za $D = -4$, $D = -3$ kubna jednadžba dana u terminima kvadratnog neostatka g jednadžbama

$$y^2 = x^3 - g^k x, \qquad k = 0, 1, 2, 3$$
$$y^2 = x^3 - g^k, \qquad k = 0, 1, 2, 3, 4, 5,$$

tim redoslijedom. Postoje 4 odnosno 6 klasa izomorfizama za krivulje s ove dvije vrijednosti D respektivno. Za ostale diskriminante D odgovarajuća krivulja i njeno kvadratno zakretanje su

$$y^2 = x^3 - 3cg^{2k}x + 2cg^{3k}, \qquad k = 0, 1$$

pri čemu je $c = j(j - 1728)^{-1} \bmod p$. Ova metoda pruža više općenitosti od tzv. rješenja zatvorene forme (što je slučaj u sljedećem algoritmu) ali je puno teža za implementaciju, uglavnom zbog toga što sadrži izračunavanje Hilbertovog polinoma klasa.
Primijetimo još jednu važnu značajku. Prije stvarnog izračunavanja parametara, mi već znamo potencijalne redove krivulje. To nam omogućava analizu potencijalnih redova prije no što se upustimo u izračunavanje parametara (a, b). Ako se dobiveni red grupe eliptičkih krivulja pokaže atraktivan za primjenu u kriptografiji, lako možemo dodatno izračunati pripadne (a, b).

Primjer 5.7.1. *Malo ćemo proći kroz izračun algoritma 5.7.1. Uzmimo neka nam je ulazni podatak jedan Mersennov prost broj,* $p = 2^{89} - 1$. *Želimo saznati potencijalne krivulje* $E_{a,b} \in (\mathbb{F}_p)$ *i njihove redove. U 4.3. koraku algoritma, traženje kvadratne forme za* $4p$,

pronalazimo mnoge reprezentacije za $4p$ *od kojih su neke:*

$$\begin{aligned}
4p &= 13462017247231^2 &&+ 3 \cdot 27656549376269^2 \\
&= 48295231754575^2 &&+ 11 \cdot 3611228302023^2 \\
&= 45656439196037^2 &&+ 43 \cdot 3016889043405^2 \\
&= 21533115172379^2 &&+ 67 \cdot 5480229090897^2 \\
&= 46660129713638^2 &&+ 24 \cdot 3527938336085^2 \\
&= 1420969356634^2 &&+ 88 \cdot 5302080342849^2 \\
&= 42554762242663^2 &&+ 267 \cdot 1578142236225^2 \\
&= 28743118396413^2 &&+ 499 \cdot 1818251501825^2.
\end{aligned}$$

Za upravo nabrojane reprezentacije, nama su od interesa diskriminante $D = -3, -11, -43, -67, -24, -88, -267, -499$ *respektivno. Naravno, postoje još brojne druge vrijednosti D koje bismo mogli koristit za ovaj p. Općenito će red odgovarajuće krivulje biti:*

$$p + 1 \pm u,$$

pri čemu je u prvi broj koji je u reprezentaciji kvadriran. Za ilustraciju detaljnijeg rada algoritma, uzmimo $D = -499$*. Sada u koraku 4.5. o paramterima krivulje dobivamo*

$$\begin{aligned}
T_{-499} = {} & 467113318239995478279867315443744131094937 6 \\
& - 606371782549426639472239256001105100 8x \\
& + 300510110807102620070672596992 0x^2 \\
& + x^3.
\end{aligned}$$

Primijetimo da je konstantni član ovog polinoma kub ($\sqrt[3]{467113318239995478279867315443744131094937 6} = 167163228389376$)*. Sljedeći korak algoritma je reduciranje polinoma modulo p.*

$$\begin{aligned}
S = T_{-499}(x) \pmod{p} = {} & 489476008241378181249146744 \\
& + 356560280230433613294194825x \\
& + 166270576558338910192101 5x^2 \\
& + x^3.
\end{aligned}$$

Sada tražimo korijen j od $S = T \bmod p$*. Dobivamo*

$$j = 43130212781604561533945186 8.$$

To je vrijednost koja "daje prvotnu iskru" konstrukciji parametara. Dobivamo

$$c = j/(j - 1728)\ mod\ p = 544175025087910210133176287.$$

Za kvadratni neostatak g u ovom prolazu algoritma je uzeto $g = -1$. *Iz toga slijede dvije verzije jednadžbe*

$$y^2 = x^3 + 22438498366433978194915747 2x \pm 46938003053313028281679046 3,$$

s redom krivulje koji im odgovara u redoslijedu kako su navedene

$$|E| = 2^{89} \pm 28743118396413.$$

Primjer 5.7.2. *Uzmimo sada neka nam je ulazni podatak* $p = 59$. *U 4.3. koraku algoritma, traženje kvadratne forme za* $4p$, *pronalazimo sljedeću reprezentaciju za* $4p$:

$$4p = 8^2 + 43 \cdot 2^2.$$

Za upravo navedenu reprezentaciju, od interesa nam je diskriminanta $D = -43$. *Sada u koraku 4.5. o paramterima krivulje dobivamo*

$$T_{-43}(x) = x + 884736000$$

Primijetimo da je konstantni član ovog polinoma kub ($\sqrt[3]{884736000} = 960$). *Sljedeći korak algoritma je reduciranje ovog polinoma modulo p.*

$$S = T_{-43}(x) \pmod{p} = x + 25$$

Sada tražimo korijen j *od* $S = T$ *mod p. Dobivamo*

$$j = -25 \equiv 34.$$

To je vrijednost koja "daje prvotnu iskru" konstrukciji parametara. Dobivamo

$$c = j/(j - 1728) \pmod{p} = 34/17 \pmod{59} = 2$$

pa je $r = -3c \pmod{p} = -6 \pmod{59} = 53$, $s = 2c\ mod\ p = 4$. *Za kvadratni neostatak g, u ovom prolazu algoritma, uzeto je* $g = 8$. *Iz toga slijede dvije verzije jednadžbe*

$$y^2 = x^3 + 53x + 4$$

i

$$y^2 = x^3 + 29x + 42.$$

s redom krivulje koji im odgovara u redoslijedu kako su navedene $|E| = 52$ *i* $|E| = 68$.

Polinomijalno računanje za veće diskriminante može biti teško. Npr. može se pojaviti problem preciznosti zbog korištenja floating-point aritmetike kod računanja Hilbertovog polinoma klasa. Zbog toga se isplati razmotriti značaj osnivanja nekih eksplicitnih parametara krivulje za male $|D|$ i na taj način izbjeći izračunavanje polinoma klasa. U tablici 5.1 nalazi se kompletna lista parametara krivulje za sve diskriminante D s $h_D = 1, 2$.

D	r	s
-7	125	189
-8	125	98
-11	512	539
-19	512	513
-43	512000	512001
-67	85184000	85184001
-163	151931373056000	151931373056001
-15	1225 - 2080 $\sqrt{5}$	5929
-20	108250 + 29835 $\sqrt{5}$	174724
-24	1757 - 494 $\sqrt{2}$	1058
-35	-1126400 - 1589760 $\sqrt{5}$	2428447
-40	54175 - 1020 $\sqrt{5}$	51894
-51	75520 - 7936 $\sqrt{17}$	108241
-52	1778750 + 5125 $\sqrt{13}$	1797228
-88	181713125 - 44250 $\sqrt{2}$	181650546
-91	74752 - 36352 $\sqrt{13}$	205821
-115	269593600 - 89157120 $\sqrt{5}$	468954981
-123	1025058304000 - 1248832000 $\sqrt{41}$	1033054730449
-148	499833128054750 + 356500625 $\sqrt{37}$	499835296563372
-187	91878880000 - 1074017568000 $\sqrt{17}$	4520166756633
-232	1728371226151263375 - 11276414500 $\sqrt{29}$	1728371165425912854
-235	7574816832000 - 190341944320 $\sqrt{5}$	8000434358469
-267	3632253349307716000000 - 12320504793376000 $\sqrt{89}$	3632369580717474122449
-403	16416107434811840000 - 4799513373120384000 $\sqrt{13}$	33720998998872514077
-427	564510997315289728000 - 5784785611102784000 $\sqrt{61}$	609691617259594724421

Tablica 5.1: Lista parametara krivulje za sve diskriminante D s $h_D = 1, 2$

Sada slijedi algoritam za računanje eksplicitnih parametara CM krivulja čiji je broj klasa 1 ili 2. Ulazni podatak algoritma je prost broj $p > 3$, a izlazni podatak je eksplicitna CM krivulja $y^2 = x^3 + ax + b$ nad $\mathbb{F}_p$. Ovdje je pretraga nad svim diskriminantama D s brojem klasa $h_D = 1, 2$ iscrpljujuća jer algoritam vraća svaki skup parametara (a, b) CM krivulja za dozvoljen broj klasa.

Algoritam 5.7.2. **Računanje eksplicitnih parametara CM krivulja s brojem klasa 1 i 2**

1. Postavimo cijelu listu diskriminanti

$$\begin{aligned}\Delta \leftarrow \{ &-3, -4, -7, -8, -11, -19, -43, -67, -163, -15, -20, -24, -35, \\ &-40, -51, -52, -88, -91, -115, -123, -148, -187, -232, -235, \\ &-267, -403, -427\}.\end{aligned}$$

2. Odaberemo idući po redu još neodabrani $D \in \Delta$ i za njega radimo:

   2.1. Pokušamo pronaći reprezentaciju za $4p = u^2 + |D|v^2$ preko Cornacchia algoritma. Ako jednadžba nema rješenja, idemo na korak 2. Ako jednadžba ima rješenja, tada nam u i v daju potencijalne redove krivulja.

   2.2. Odaberemo na slučajan način kvadratni neostatak $g \pmod p$. Ako vrijedi $p \equiv 1 \pmod 3$ i $g^{(p-1)/3} \equiv 1 \pmod p$, idemo na korak 2.2. Ako je $D = -3$, i uz spomenute uvjete je još i g kub modulo p, idemo na korak 2.2.

   2.3. Ako je $D = -3$, postavimo *parametri* $\leftarrow \{(0, -g^k) : k = 0, 1, 2, 3, 4, 5\}$. Na taj način dobijemo šest krivulja oblika $y^2 = x^3 - g^k$.
   Ako je $D = -4$, postavimo *parametri* $\leftarrow \{(-g^k, 0) : k = 0, 1, 2, 3\}$. Na taj način dobijemo četiri krivulje oblika $y^2 = x^3 - g^k x$.
   Inače, odaberimo par (r, s) iz tablice 5.1. Postavimo *parametri* $\leftarrow \{(-3rs^2g^{2k}, 2rs^5g^{3k}) : k = 0, 1\}$. Pripadne dvije krivulje će biti oblika $y^2 = x^3 - 3rs^3g^{2k}x + 2rs^5g^{3k}$.

3. Vratimo *parametri*.

Parametri su u ovom algoritmu mogli biti izračunati i preko Hilbertovog polinoma klasa. Međutim, dobivanje parametara ovako eksplicitnim putem znači da se izgradnja CM krivulja može postići veoma brzo uz minimum programiranja. Čak nije nužna niti provjera vrijedi li $4a^3 + 27b^2 \neq 0$, najbitniji zahtjev za pravovaljane eliptičke krivulje nad $\mathbb{F}_p$. Još jedno zanimljivo obilježje je da su tabelirani parameteri r, s skloni imati zanimljivu faktorizaciju. Ustvari parametri s su najčešće glatki brojevi. U poglavlju o faktorizaciji eliptičkih krivulja se nalazi definicija prirodnog broja koji je B-gladak.

Slijedi implementacija algoritma 5.7.2. u programskom jeziku Python.

```
from Cornacchia import Cornacchia
import random
from math import pow, sqrt

D = "-3, -4, -7, -8, -11, -19, -43, -67, -163, -15, -20, -24,
 -35, -40, -51, -52, -88, -91, -115, -123, -148, -187, -232,
 -235, -267, -403, -427".split(", ")

tablica = [
    [-7, 125, 189],
    [-8, 125, 98],
    [-11, 512, 539],
    [-19, 512, 513],
    [-43, 512000, 512001],
    [-67, 85184000, 85184001],
    [-163, 151931373056000, 151931373056001],
    [-15, 1225 - 2080*sqrt(5), 5929],
    [-20, 108250 + 29835*sqrt(5), 174724],
    [-24, 1757 - 494*sqrt(2), 1058],
    [-35, -1126400 - 1589760*sqrt(5), 2428447],
    [-40, 54175 - 1020*sqrt(5), 51894],
    [-51, 75520 - 7936*sqrt(17), 108241],
    [-52, 1778750 + 5125*sqrt(13), 1797228],
    [-88, 181713125 - 44250*sqrt(2), 181650546],
    [-91, 74752 - 36352*sqrt(13), 205821],
    [-115, 269593600 - 89157120*sqrt(5), 468954981],
    [-123, 1025058304000 - 1248832000*sqrt(41), 1033054730449],
    [-148, 499833128054750 + 356500625*sqrt(37), 499835296563372],
    [-187, 91878880000 - 1074017568000*sqrt(17), 4520166756633],
    [-232, 17283712261512633 75 - 11276414500*sqrt(29),
    17283711654259128 54],
    [-235, 7574816832000 - 190341944320*sqrt(5), 8000434358469],
    [-267, 3632253349307716000000 - 1232050479337600 0*sqrt(89),
    3632369580717474122449],
    [-403, 16416107434811840000 - 47995133731203840 00*sqrt(13),
    33720998998872514077],
    [-427, 564510997315289728000 - 57847856111027840 00*sqrt(61),
    609691617259594724421],
```

```
]

def kvadratni_neostatak(p):
    neostatci = list();
    for d in range(1, p):
        postoji_rjesenje = False
        for x in range(1, p):
            if ((x * x - d) % p == 0):
                postoji_rjesenje = True
        if not postoji_rjesenje: neostatci.append(d);
    return neostatci

global parametri, red

def algoritam(p):
    global parametri, red
    parametri = list(); red = list()
    for i in range(0, len(D)):
        d = int(D[i]); d *= -1
        rjesenje = Cornacchia(d, p)
        if rjesenje == "Jednadzba nema rjesenja":
            continue
        neostatci = kvadratni_neostatak(p);
        g = random.choice(neostatci)
        u = rjesenje['u']; v = rjesenje['v']
        x_u = 2 * u; y_v = 2 * v
        while (p-1)%3 == 0 and (pow(g,(p-1)/3)-1)%p == 0:
            if D[i] == "-3" and not(pow(g, 3) % p == 0):
                break
            neostatci = kvadratni_neostatak(p)
            g = random.choice(neostatci)
        if D[i] == "-3":
            print("D je "+ D[i] + ", a g je: " + str(g))
            for k in range (0, 6):
                x = 0; y = -pow(g, k)
                parametri.append((int(x), int(y)))
            red.append(p + 1 + x_u)
            red.append(p + 1 - x_u)
            red.append(int(p + 1 + (x_u + 3 * y_v) / 2))
```

```
            red.append(int(p + 1 + (x_u - 3 * y_v) / 2))
            red.append(int(p + 1 - (x_u + 3 * y_v) / 2))
            red.append(int(p + 1 - (x_u - 3 * y_v) / 2))
        elif D[i] == "-4":
            print("D je " + D[i] + ", a g je " + str(g))
            for k in range (0, 4):
                x = -pow(g, k); y = 0
                parametri.append((int(x), int(y)))
            red.append(p + 1 + x_u)
            red.append(p + 1 - x_u)
            red.append(int(p + 1 + 2 * y_v))
            red.append(int(p + 1 - 2 * y_v))
        else:
            for redak in tablica:
                if str(redak[0]) == D[i]:
                    r = redak[1]; s = redak[2]
                    break
            print("D je "+ D[i] + ", a g je " + str(g))
            for k in range (0, 2):
                x = -3*r*s*s*s*(g**(2*k))
                y = 2*r*(s**5)*(g**(3*k))
                parametri.append((x, y))
            red.append(p + 1 + x_u)
            red.append(p + 1 - x_u)
    parametri_red = zip(parametri, red)
    return parametri_red

if __name__ == '__main__':
    #h = algoritam(17)
    h = algoritam(103)
    p, r = zip(*zip(parametri, red))
    print(p)
    print(r)
```

Kratko ćemo ilustrirati rad algoritma za dva proizvoljna primjera.

Primjer 5.7.3. *Uzmimo neka nam je ulazni podatak prost broj* $p = 17$. *U koraku 2.1. nalazimo da je dotična jednadžba rješiva za dvije diskriminante s brojem klasa manjim ili jednakim dva. To su diskriminante :* -4 *i* -8*, a pripadne im paramtere i redove koje nam algoritam daje ćemo prikazati u donjoj tablici. Za prolaz algoritma s diskriminantom* -4 *u koraku 2.2. je odabran broj* 3 *za kvadratni neostatak g, dok je u prolazu s diskriminantom* -8 *odabran broj* $g = 5$. *Algoritam daje listu parametara* $(a, b) = [(-1, 0), (-3, 0), (-9, 0), (-27, 0), (-352947000, 2259801992000), (-8823675000, 282475249000000)]$ *i listu mogućih redova krivulja* $[20, 16, 22, 14, 24, 12]$. *Primjetimo samo još da nam algoritam ne kaže koji par parametara ide s kojim redom krivulje, ali to i nije neki problem. Jednostavno pronađemo točku P na krivulji i ako je kandidat za red grupe pripadne eliptičke krivulje ne poništi, možemo biti sigurni da taj kandidat pripada nekoj drugoj krivulji. U tablici će svakom paru parametara biti pridodan red koji mu pripada. Također će parametri (a,b) u tablici biti prikazani u njima ekvivalentnim brojevima modulo p.*

D	E	$\lvert E(\mathbb{F}_p)\rvert$
-4	$y^2 = x^3 + 16x + 0$	16
	$y^2 = x^3 + 14x + 0$	26
	$y^2 = x^3 + 8x + 0$	20
	$y^2 = x^3 + 7x + 0$	10
-8	$y^2 = x^3 + 13x + 3$	12
	$y^2 = x^3 + 2x + 1$	24

Primjer 5.7.4. *Uzmimo neka nam je ulazni podatak prost broj* $p = 103$. *U koraku 2.1. nalazimo da je dotična jednadžba rješiva za tri diskriminante s brojem klasa manjim ili jednakim dva:* -3*,* -11 *i* -67*, a pripadne im paramtere i redove koje nam algoritam daje ćemo prikazati u donjoj tablici. Za prolaz algoritma s diskriminantom* -3 *u koraku 2.2. je odabran broj* 86 *za kvadratni neostatak g, za prolaz s diskriminantom* -11 *je odabran* $g = 73$ *dok je u prolazu s diskriminantom* -67 *odabran broj* $g = 45$. *Algoritam daje listu parametara* $(a, b) = [(0, -1), (0, -86), (0, -7396), (0, -636056), (0, -54700816), (0, -4704270176), (-240523497984, 46584751438539776), (-1281749720756736, 1812226025036642804019 2), (-157962277893429387100360959552000, 764149261810087780722661738894877473218730368000), (-319873612734194508878230943092800000, 6963310148244424901835255095679570974705680478400000 0)]$ *i listu mogućih redova krivulja* $[124, 84, 117, 111, 91, 97, 108, 100, 116, 92]$. *Slično kao u prethodnom primjeru,*

sada slijedi tablica s diskriminantama i krivuljama te njima pripadnim redovima.

D	E	$\lvert E(\mathbb{F}_p)\rvert$
-3	$y^2 = x^3 + 0x + 102$	124
	$y^2 = x^3 + 0x + 17$	111
	$y^2 = x^3 + 0x + 20$	91
	$y^2 = x^3 + 0x + 72$	84
	$y^2 = x^3 + 0x + 12$	97
	$y^2 = x^3 + 0x + 2$	117
-11	$y^2 = x^3 + 95x + 85$	108
	$y^2 = x^3 + 10x + 46$	100
-67	$y^2 = x^3 + 1x + 41$	116
	$y^2 = x^3 + 68x + 6$	92

Poglavlje 6

Zaključak

Jedna od prednosti **CM metode** se očituje u njenom redoslijedu izvođenja radnji prilikom generiranja eliptičke krivulje. Prvo se odabire red grupe eliptičkih krivulja modulo p koji zadovoljava tri glavna sigurnosna uvjeta, a potom konstruira krivulja s tim redom. Za dani prost broj p, Atkin-Morain metoda i sve njene varijante računaju j-invarijantu krivulje, te saznaju red m i parametre (a, b) takve da je zadovoljeno $m = |E_{a,b}(\mathbb{F}_p)|$. Vrijeme izvođenja tih algoritama je $O((\log p)^5)$. Uzmemo li u obzir provjeru prostosti broja p i sigurnosnu provjeru broja m, uključujući vjerojatnost njihovog uspjeha, vrijeme izračunavanja pogodne trojke $(m, (a, b), p)$ dobivene na temelju CM metode je $O((\log p)^6)$. Stoga je CM metoda zasnovana na Atkin-Morain algoritmu puno učinkovitija od metode brojanja točaka eliptičke krivulje zasnovane na drugim algoritmima. Npr. vrijeme izračunavanja pogodne trojke $(m, (a, b), p)$ dobivene na osnovu poboljšanog Schoofovog algoritma je $O((\log p)^7)$, što je sporije od dobivanja pogodne trojke na osnovu CM metode.

Sljedeća bitna značajka očituje se u jednakoj ili čak bolje osiguranoj otpornosti na poznate napade. Najpoznatiji algoritmi za napad na ECDLP imaju eksponencijalno vrijeme ovisno o veličini grupe eliptičke krivulje, dok za konvencijalne (ne-eliptičke) grupe generirane prostim brojevima algoritmi za napad na DLP imaju subeksponencijalno vrijeme. To povlači da u kriptosustavima nad eliptičkim krivuljama možemo postići jednak stupanj sigurnosti kao kod RSA ili DSA algoritama uz korištenje manjih parametara. Posljedica korištenja manjeg ključa se očituje u bržoj implementaciji, potrebi za manjom količinom prostora za pohranu od dosadašnje i smanjenim zahtjevima na bandwith (količinu informacija koja može biti prenesena u određenom vremenskom periodu preko žičnog ili bežičnog linka). Ova prednost je ključna kada nas zanima implementacija na ograničenim uređajima kao što su pametne kartice, mobiteli, radijski prijemnici ili senzori. Odnos duljina ključeva između kriptosustava eliptičkih krivulja te najkorištenijeg i najpopularnijeg kriptosustava s javnim ključem - RSA kriptosustava je prikazan u tablici 6.1. Razlika u potrebnoj veličini

ključa za jednak stupanj sigurnosti je i više nego očita.

ECC	**RSA**
106 bitova	512 bitova
132 bita	768 bitova
160 bitova	1024 bita
224 bita	2048 bitova
256 bitova	3072 bita
384 bita	7680 bitova
521 bit	15360 bitova

Tablica 6.1: Odnos duljine ključeva za različite kriptosustave

Još ćemo za spomenute vrste kriptosustava prikazati potrebno vrijeme za generiranje i provjeru potpisa njhovim algoritmima. Primjetimo da kada udvostručimo duljinu ključa kod RSA, potrebno vrijeme generiranja postaje skoro 5 puta veće, dok je kod ECC to povećanje neznatno veće od dvostrukog. Još jedan primjer koji "ide na ruku" kriptosustavu elitpičkih krivulja.

	Generiranje potpisa	**Provjera potpisa**
RSA; ključ: 1024 bita	25 ms	< 2 ms
ECC; ključ: 160 bitova	32 ms	33 ms
RSA; ključ: 2048 bitova	120 ms	5 ms
ECC; ključ: 216 bitova	68 ms	70 ms

Tablica 6.2: Vrijeme generiranja i provjere potpisa za pojedini kriptosustav i duljinu ključa

Kao jedan primjer iz prakse možemo navesti korištenje eliptičkih krivulja kod enkripcije i zaštite internet protokola. Koriste se kriptosustavi eliptičkih krivulja zbog svih dosada navedenih osobina, ali i zbog izrazito teške faktorizacije velikog složenog broja na proste faktore. Ilustrativno, prosti brojevi su na neki način blokovi koji izgrađuju ogromne složene brojeve koji leže u pozadini enkripcije interneta, a njihova faktorizacija je mukotrpan i veoma dugotrajan proces. Osim toga, vremenom će korisnici postajati sve osjetljiviji na vlastitu on-line privatnost pa će možda zahtijevati zaštitu za više svojih transakcija. Npr. Yahoo korisnici bi mogli zahtijevati opciju zaštite pristupanja emailu umjesto dosadašnje zaštite samo pomoću prijave zaporkom. Slično, ljubitelji knjiga bi mogli zahtijevati zaštitu privatnosti njihovih navika posudbe knjiga s Amazona. Primjera je pregršt, a svi nas navode na jedan zaključak: moramo biti spremni korisnicima osigurati svu privatnost koju bi oni

mogli zahtijevati i to im moramo omogućiti najučinkovitijim postojećim kriptosustavima. Vjeruje se da porast ovakve vrste zahtjeva i potreba za sve većom privatnošću predviđaju široku implementaciju kriptosustava eliptičkih krivulja ne samo u bežičnim okruženjima već i na desktop/server okruženjima. Svojstva eliptičkih krivulja i prednost metoda zasnovanih na kompleksnom množenju su spominjani kroz cijeli rad. Jedino što možemo zaključiti bez previše ponavljanja već napisanog jest to da će u budućnosti postojati sve veća potreba za korištenjem eliptičkih krivulja, kriptosustava zasnovanih na njima, i matematike kao podloge za računarske i srodne znanosti.

Bibliografija

[1] I. F. Blake, G. Seroussi i N. P. Smart, *Elliptic Curves in Cryptography*, Cambridge University Press, 2002.

[2] R. Crandall i C. Pomerance, *Prime Numbers: A Computational Perspective*, Springer Science+Business Media and Inc., 2005.

[3] A. Dujella, *Eliptičke krivulje u kriptografiji*, PMF-MO, 2013.

[4] H. Eberle, S. Fung, V. Gupta, N. Gura, S.C. Shantz i D. Stebila, *Speeding up Secure Web Transactions Using Elliptic Curve Cryptography*, (2017), https://www.isoc.org/isoc/conferences/ndss/04/proceedings/Papers/Gupta.pdf.

[5] D. Hankerson, A. Menezes i S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer-Verlag New York and Inc., 2004.

[6] M. Khalgui, O. Mosbahi i A. Valentini, *Embedded Computing Systems: Applications, Optimization, and Advanced Desgin*, IGI Global, 2013.

[7] K. Y. Lam, E. Okamoto i C. Xing, *Advances in Cryptology - ASIACRYPT' 99*, Springer-Verlag Berlin Heidelberg, 1999.

Sažetak

Ovaj rad proučava metodu generiranja eliptičkih krivulja pomoću kompleksnog množenja čiji je cilj proizvesti što otporniju krivulju na razne vrste kriptografskih napada. Rad se može promatrati kroz tri osnovne cjeline. Prvi dio rada obuhvaća prva četiri poglavlja i pokriva aritmetiku Weierstrassovih eliptičkih krivulja.

Drugi dio rada smatramo glavnim dijelom rada. Riječ je o poglavlju 5 u kojem obrađujemo generiranje "sigurnih" parametara za eliptičku krivulju koristeći CM metodu. Također pokrivamo dva različita redoslijeda izvođenja radnji prilikom generiranja eliptičkih krivulja i zaključujemo kako je bolje prvo odabrati prost broj p, a zatim fundamentalnu diskriminantu $-D$. Prilikom generiranja eliptičke krivulje, potrebno je izračunati minimalni polinom nekog generatora Hilbertovog polja klasa. Možemo izračunati Hilbertov polinom ili Weberov polinom čije smo prednosti prilikom korištenja već objasnili.

Treći dio rada se odnosi na šesto poglavlje odnosno zaključak u kojem iznosimo cjelokupni dojam obrađene teme. Obrađeni algoritmi u ovom radu su uglavnom potkrijepljeni primjerima kojima je ilustrirano njihovo izvođenje. Kroz primjere smo nastojali čitatelju ovog rada približiti i na konkretnim brojevima prikazati redoslijed izvođenja koraka algoritma, te tako omogućiti lakše razumijevanje teoretskog dijela ovog rada.

Summary

This thesis studies a generation method of elliptic curves using complex multiplication with a purpose to create a curve that is as resistant as possible of different types of cryptographic attacks. It may be divided into three fundamental parts. The first part includes the first four chapters and covers arithmetic of Weierstrass elliptic curves.

The second part (Chapter 5) is the main part of the thesis. The Chapter 5 describes the generating of „secure“ elliptic curve parameters using the CM method. Two different executing orders of generating of elliptic curves are also covered and the conclusion is that it is better to choose the prime number p first and the fundamental discriminant $-D$ second. During generating of an elliptic curve, it is necessary to compute minimal polynomial of the Hilbert class field generator. We can compute the Hilbert polynomial or Weber polynomial with already described advantages.

The third part of the thesis refers to the sixth chapter apropos conclusion where the entire impression of the elaborated topic is described. Processed algorithms in this thesis are mostly confirmed with examples which illustrate their execution. The examples are in the thesis to bring the algorithms closer to the reader and to show algortihm execution on concrete numbers in purpose to enable easier understanding of the theoretical part of the thesis.

Životopis

Rođena sam 24.veljače 1993. u Splitu. Završila sam Osnovnu školu "Jesenice" na Oriju i Osnovnu glazbenu školu "Lovro pl. Matačić" u Omišu, a zatim Jezičnu gimnaziju u Srednjoj školi "Jure Kaštelan" u Omišu. 2011. godine sam upisala Preddiplomski sveučilišni studij Matematika na Matematičkom odsjeku PMF-a u Zagrebu. Preddiplomski sveučilišni studij sam završila 2014. te stekla titulu univ.bacc.math. U listopadu 2014. godine upisujem Diplomski sveučilišni studij Računarstvo i matematika na PMF-u u Zagrebu kojeg trenutno završavam.