UNIVERZA V LJUBLJANI

FAKULTETA ZA MATEMATIKO IN FIZIKO

Matematika - izobraževalna smer

Erik Schlegel

POLLARDOVA ρ-METODA

Magistrsko delo

Ljubljana, 2003

To delo posvečam svojim staršem,
ki so me ves čas študija
spodbujali in mi
stali ob strani.

Zahvaljujem se svojemu mentorju prof. dr. Aleksandru Jurišiću za nasvete in pomoč pri nastajanju magistrskega dela.

Izjavljam, da sem avtor tega magistrskega dela.

Erik Schlegel

Kazalo

PROGRAM MAGISTRSKEGA DELA

Pollardova ρ-metoda

Delo naj predstavi matematične osnove, ki so potrebne za razumevanje in implementacijo Pollardove ρ-metode, s poudarkom na računanju diskretnega logaritma. Glavni cilji tega dela naj bodo podroben opis Pollardove ρ-metode, analiza kompleksnosti in praktičen pomen tega algoritma za kriptosistem z eliptičnimi krivuljami.

Literatura:

J. M. Pollard, *Monte Carlo methods for index computation* (mod p), Mathematics of Computation, **32** (1978), 918-924.

D. R. Stinson, *Cryptography: Theory and Practice, Second Edition*, CRC Press, 2002.

A. J. Menezes, P. C. Van Oorschot in S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

Certicom Corp., (`http://www.certicom.ca/resources/ecc_chall/`).

E. E. Teske, *New Algorithms for Finite Abelian Groups*, Ph.D. Thesis, Technische Univerität Darmstadt, 1998.

Mentor: prof. dr. Aleksandar Jurišić

Povzetek

Delo predstavlja Pollardovo ρ-metodo za reševanje matematičnih problemov, s poudarkom na reševanju problema diskretnega logaritma in problema diskretnega logaritma na eliptični krivulji nad končnim obsegom. Glavni namen tega dela je predstavitev izreka o asimptotični zgornji meji časovne in prostorske zahtevnosti Pollardove ρ-metode.

Ključne besede: kriptografija, problem diskretnega logaritma, eliptične krivulje, Pollardova ρ-metoda, paradoks rojstnega dne, Floydov algoritem iskanja cikla, Brentov algoritem iskanja cikla.

Abstract

This thesis deals with Pollard's rho method for solving computational problems, especially for solving discrete logarithm problem on elliptic curves over finite field. The main goal is to represent a theorem, which gives an asymptotic upper bound on time and space complexity of the Pollard's rho method.

Keywords: cryptography, discrete logarithm problem, elliptic curves, Pollard's rho method, birthday paradox, Floyd's cycle-finding algorithm, Brent's cycle-finding algorithm.

Mathematics Subject Classification (2000): 94A60, 68W20, 68W40, 11Y16.

Tabele

Slike

Uvod

Problem računanja diskretnega logaritma je osnova varnosti številnih kriptosistemov z javnimi ključi. V vsakem takem kriptosistemu se namreč šifriranje in dešifriranje zreducira na potenciranje v neki končni ciklični podgrupi. Če gledamo na eksponent v potenci a^b v dvojiškem zapisu oziroma kot vrednost polinoma v točki 2, ki jo lahko izračunamo s Hornerjevim algoritmom, se računanje potence a^b prevede na postopno kvadriranje in množenje z a, kar pomeni, da je potenciranje izvedljivo v polinomskem času. Diskretnega logaritma, ki je inverzna operacija potenciranju, pa v polinomskem času v splošnem ne moremo izračunati, vendar imamo za njegovo računanje na voljo številne algoritme. Preden začnemo s pregledom najbolj znanih izmed njih, bomo natančno definirali problem diskretnega logaritma, ki ga označujemo s kratico **DLP** (angl. Discrete Logarithm Problem).

Naj bo G končna grupa, pisana multiplikativno, in g njen element reda n, tj. n je najmanjše tako naravno število, da je $g^n = 1$. Potem je

$$\langle g \rangle = \{g^i \,:\, 0 \leq i \leq n-1\} \tag{1}$$

ciklična podgrupa grupe G reda n, tj. $|\langle g \rangle| = n$. Naj bo h poljuben element iz podgrupe $\langle g \rangle$. **Problem diskretnega logaritma** v ciklični podgrupi $\langle g \rangle$ grupe G je enačba

$$g^x = h.$$

Njeno rešitev x imenujemo **diskretni logaritem elementa h v osnovi g**. Pišemo $x = \log_g h$. Računanje diskretnega logaritma s pomočjo logaritemske tabele je v primejavi s takim računanjem običajnega logaritma v realnih številih za velike n nepraktično, zato si pomagamo z drugačnimi metodami. Na primer, vsak primer DLP v grupi $\langle g \rangle$ reda n lahko rešimo z uporabo metode računanja

zaporednih potenc generatorja $g, g^2, g^3, \ldots$, dokler ne pridemo do tistega elementa, katerega diskretni logaritem iščemo. Tak način imenujemo **metoda grobe sile** (angl. brute force algorithm ali exhaustive search), glej Stinson [30, str. 229]. Njegova časovna zahtevnost je $O(n)$, prostorska zahtevnost pa $O(1)$, ker izračunamo vsako naslednjo potenco generatorja tako, da prejšnjo potenco pomnožimo z generatorjem g. Običajno izrazimo zahtevnost algoritma v odvisnosti od števila bitov, ki jih zasedajo vhodni podatki. Na primer, za zapis števila n potrebujemo $m = \lfloor \log_2 n \rfloor + 1$ bitov. V tem smislu je metoda grobe sile eksponentne časovne zahtevnosti, saj je $O(n) = O(2^m)$. Če je n "majhno" število, je uporaba metode grobe sile mogoča. Na primer, v ciklični grupi reda do 40 bitov, tj. za $n < 10^{12}$. Če pa je n večje število, pa postane tak način računanja diskretnega logaritma računsko prezahteven. Vendar si lahko v večjih grupah pomagamo z boljšimi algoritmi. Najbolj znani med njimi so

- **metoda veliki-mali korak**,
- **Pollardova ρ-metoda**,
- **metoda index calculus**.

Časovna zahtevnost metode veliki-mali korak in Pollardove ρ-metode je $O(\sqrt{n}\,) = O(2^{m/2})$. Prednost Pollardove ρ-metode pred metodo veliki-mali korak je v tem, da lahko Pollardovo ρ-metodo implementiramo s konstantno prostorsko zahtevnostjo, prostorska zahtevnost metode veliki-mali korak pa je $O(\sqrt{n}\,)$. Čeprav imata takó metoda veliki-mali korak kot tudi Pollardova ρ-metoda še vedno eksponentno časovno zahtevnost, sta bistveno boljši od metode grobe sile. Z njima lahko npr. rešimo vsak primer DLP v ciklični grupi reda do 80 bitov, tj. za $n < 10^{24}$. V nekaterih še večjih grupah pa lahko uporabimo metodo index calculus, ki je podeksponentne časovne in prostorske zahtevnosti. Z metodo index calculus lahko npr. rešimo vsak primer DLP v ciklični grupi $(\mathbb{Z}_p^*, *)$, kjer je p praštevilo, reda do 500 bitov, tj. za $p < 10^{150}$. Vendar metode index calculus za reševanje DLP v grupi na eliptični krivulji nad končnim obsegom zaenkrat ne znamo uporabiti. Ker je Pollardova ρ-metoda za te grupe trenutno najboljša, je cilj tega dela njena natančna predstavitev.

Magistrsko delo je organizirano takole: V 1. poglavju so predstavljene kriptografija z javnimi ključi, varnost diskretnega logaritma in eliptične krivulje. V

2. poglavju so opisani naslednji algoritmi za reševanje DLP: metoda veliki-mali korak, metoda index calculus in Pohlig-Hellmanov algoritem. V 3. poglavju je predstavljena Pollardova ρ-metoda. Le-ta temelji na oblikovanju periodičnega zaporedja $\{y_i\}$, ki ga imenujemo **Pollardovo zaporedje**, in na iskanju takih $i, j \in \mathbb{N}$, $i \neq j$, da je $y_i = y_j$, tj. na iskanju **trčenja**. Elementi Pollardovega zaporedja se shranjujejo v tabelo, vendar pa se je temu možno izogniti s primernim algoritmom iskanja trčenja. Zato so v 4. poglavju predstavljeni naslednji algoritmi iskanja trčenja: Floydov algoritem, Brentov algoritem in algoritem primerjaj in uredi. Najpogosteje uporabljene grupe v kriptografiji so multiplikativne grupe končnih obsegov in grupe na eliptičnih krivuljah nad končnimi obsegi. Zato so v 5. poglavju predlagana pravila za oblikovanje Pollardovih zaporedij v omenjenih grupah. V 6. poglavju je predstavljeno reševanje DLP v grupi na eliptični krivulji nad končnim obsegom v praksi. Zadnje poglavje pa je povzetek predstavljenih algoritmov za reševanje DLP.

Poglavje 1

Kriptografija z javnimi ključi

Kriptografijo z javnimi ključi pogosto označujemo z izrazom moderna kriptografija. Da jo lahko razumemo, moramo poznati vzroke, ki so narekovali njen razvoj, in koncept njenega delovanja. Vse to bo na kratko predstavljeno v prvem razdelku. V drugem razdelku bosta obravnavana varnost in diskretni logaritem. V kriptografiji z javnimi ključi so se grupe na eliptičnih krivuljah nad končnimi obsegi pokazale za zelo uporabne v mnogih okoljih z omejenim prostorom. Zato bodo v zadnjem razdelku definirane eliptične krivulje in DLP v grupi na eliptični krivulji nad končnim obsegom.

1.1 Koncept kriptografije z javnimi ključi

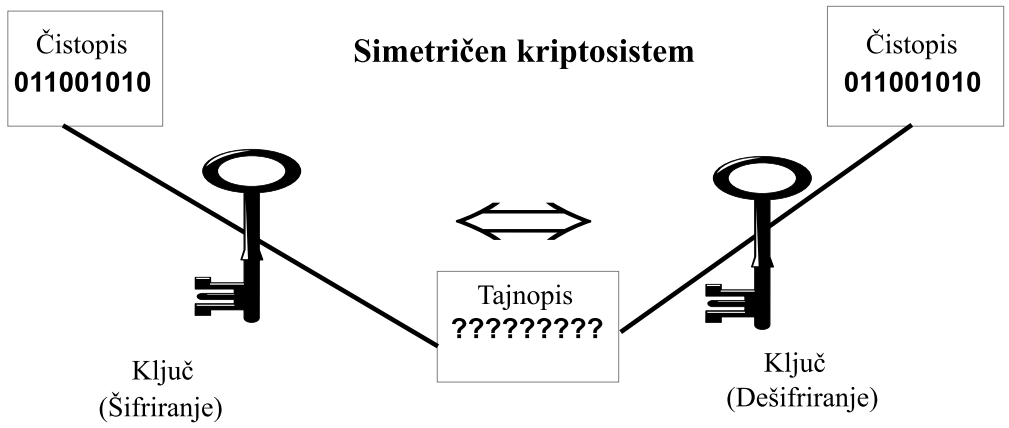
Veliko časa je bila kriptografija izključno v lasti vladnih tajnih služb in vojaških obveščevalnih organizacij, glej Singh [29, str. 279-292]. Decembra 1997 je namreč angleška vlada predstavila dokumente z opisom koncepta kriptografije z javnimi ključi[1] in RSA kriptosistma[2]. Kljub temu se v kriptografiji smatra leto 1976 za začetek razvoja kriptografije z javnimi ključi, ker sta v tem letu W. Diffie in M. Hellman objavila revolucionarni koncept kriptografije z javnimi ključi [7]. V tem razdelku bomo na kratko predstavili kriptografijo s tajnimi in z

[1] James H. Ellis je januarja 1970 predstavil kriptografijo z javnimi ključi [9]. Bil je član organizacije CESG (Communication Electronics Security Group), ki je posebna enota britanske vladne organizacije GCHQ (Government Communications Headquarters).

[2] Danes je znano, da je Clifford C. Cocks že leta 1973 razvil kriptosistem z javnimi ključi, ki je danes poznan pod imenom RSA kriptosistem [4]. Tudi on je bil član britanske vladne organizacije CESG.

javnimi ključi.

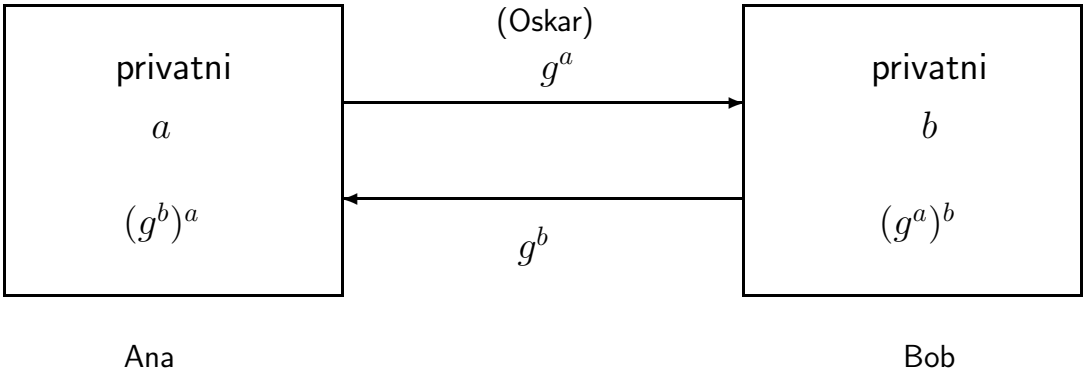
Najstarejša in hkrati tudi ena od osnovnih nalog kriptografije je omogočanje varne komunikacije po nezaščitenih poteh oziroma komunikacijskih kanalih. V splošnem potekajo vse komunikacije po poteh, ki niso varne. Danes najbolj razširjen primer take poti je mednarodno omrežje internet. Klasična rešitev problema varne komunikacije po nezaščitenem kanalu je uporaba kriptosistema s tajnimi ključi, glej Stinson [30, §1 in §3]. V vsakem takem kriptosistemu opravimo šifriranje in odšifriranje z istim tajnim ključem, glej sliko 1.1. Vsak tak ključ mora biti v strogi tajnosti, če ne želimo, da bi nezaželena oseba brez velikega truda šifrirala in odšifrirala sporočila. Kriptosisteme s tajnimi ključi imenujemo tudi simetrični kriptosistemi oziroma **simetrične šifre** (angl. symmetric ciphers). O tajnem ključu se je potrebno pred pričetkom varne komuni-



Slika 1.1: Simetričen kriptosistem.

kacije varno dogovoriti. Če to opravijo sami uporabniki, s tem ni niti nič narobe. Če pa bi morala za to skrbeti kaka centralna agencija, bi bilo to občutno preveč, glej Jurišić in Tonejc [14]. Dokler je uporabnikov simetričnega kriptosistema relativno malo, je dogovor o skupnem tajnem ključu "majhen" problem. Problem postane "večji", ko se število le-teh poveča. V primeru n uporabnikov potrebujemo $\binom{n-1}{2} = O(n^2)$ tajnih ključev in zato prav toliko tajnih dogovorov, da lahko vsak par varno komunicira z vsakim. Hkrati pa potrebuje vsak uporabnik $O(n)$ prostora za upravljanje s tajnimi ključi. Če uporabljamo npr. 128-bitne tajne

ključe, potem lahko na pametno kartico (angl. smart card) z 8Kb spominom[3] shranimo le 512 takih ključev. Prvo elegantno rešitev za izmenjavo/dogovor o skupnem (tajnem) ključu sta predlagala Diffie in Hellman leta 1976, ki je danes poznana pod imenom Diffie-Hellmanov protokol, na kratko DH protokol [7]. Ta dogodek je pomenil začetek razvoja kriptografije z javnimi ključi. Diffie in Hellman sta razvila koncept izmenjave ključev na osnovi težavnosti DLP v končni ciklični grupi, glej sliko 1.2. Uporabnika Ana in Bob, ki želita komunicirati pri-



Slika 1.2: Diffie-Hellmanov protokol.

krito z uporabo kakega simetričnega kriptosistema, npr. z AES (angl. Advanced Encryption Standard), glej Stinson [30, §3.6], ali z IDEA (angl. International Data Encryption Algorithm), glej Menezes et al. [23, §7.6], se lahko dogovorita o skupnem (tajnem) ključu po Diffie-Hellmanovemu protokolu. Najprej si izbereta ciklično grupo G reda n, generirano z elementom g, in nato vsak zase neko naključno število iz intervala $[1, n-1]$. Označimo število, ki ga je izbrala Ana z $\boldsymbol{a}$, število, ki ga je izbral Bob pa z $\boldsymbol{b}$. Števili a in b imenujemo **privatna** ali **zasebna** ključa Ane oziroma Boba. Ana izračuna element $\boldsymbol{g^a}$ in ga pošlje Bobu po javnem kanalu, medtem ko Bob izračuna element $\boldsymbol{g^b}$ in ga pošlje Ani. Elementa g^a in g^b sta **javna** ključa Ane in Boba. Tako lahko izračunata (vsak zase) element

$$g^{a\,b} = (g^a)^b = (g^b)^a$$

iz grupe G, ki postane njun skupen (tajni) ključ. Oskar, ki je prisluškovalec

[3]V mobilni telefoniji GSM uporabljamo tako imenovane "SIM kartice" (angl. Subscriber Identity Modul). To so pametne kartice z 8Kb spominom za shranjevanje ključev, telefonskih številk in podatkov o uporabniku, glej Jurišić in Tonejc [14].

omrežja, lahko posname izmenjavo ključev med Ano in Bobom, torej pozna g, g^a in g^b, vendar v splošnem to ni dovolj za izračun njunega skupnega (tajnega) ključa $g^{a\,b}$. Učinkovit algoritem za izračun diskretnega logaritma bi pomenil konec varne uporabe DH protokola. Če bi bil DLP v grupi G učinkovito rešljiv, bi lahko Oskar izračunal števili a ali b iz g^a oziroma g^b in tako tudi skupen (tajni) ključ $g^{a\,b}$ Ane in Boba. Označimo problem iskanja skupnega (tajnega) ključa $g^{a\,b}$ v DH protokolu pri znanih g, g^a in g^b z **DHP** (angl. Diffie-Hellman Problem). Domneva se, da sta DHP in DLP ekvivalentna problema[4], tj. iz rešitve enega lahko izpeljemo rešitev drugega v polinomskem času.

Kriptosisteme z javnimi ključi imenujemo tudi asimetrični kriptosistemi oziroma **asimetrične šifre** (angl. asymmetric ciphers). V njih uporabljamo različne ključe za šifriranje in odšifriranje oziroma za podpisovanje in preverjanje podpisa nekega sporočila. Šifriranje sporočila opravimo s prejemnikovim javnim ključem tako, da ga lahko odšifrira samo prejemnik s svojim zasebnim ključem. Postopek šifriranja mora zagotavljati, da lahko samo prejemnik, kateremu je sporočilo namenjeno, le-tega odšifrira in prebere. Pri digitalnem podpisu je proces šifriranja in odšifriranja zamenjan. Pošiljatelj opravi digitalni podpis s svojim zasebnim ključem tako, da lahko vsak prejemnik podpisa le-tega preveri s pošiljateljevim javnim ključem. Postopek podpisovanja mora zagotavljati, da lahko podpis preveri vsak, toda samo lastnik zasebnega ključa lahko sporočilo podpiše.

Diffie in Hellman sta leta 1976 poleg protokola za izmenjavo/dogovor o skupnem (tajnem) ključu predstavila tudi koncept kriptografije z javnimi ključi [7], vendar pa nista našla konkretnega modela za njegovo realizacijo. To se je zgodilo šele leta 1978 z iznajdbo RSA kriptosistema, katerega varnost temelji na tako imenovanem RSA problemu, ki je v tesni zvezi s problemom faktorizacije sestavljenega števila, glej Menezes et al. [23, §3.3]. Zato je postal RSA prvi praktično uporaben kriptosistem z javnimi ključi. Za prvo praktično uporabo problema diskretnega logaritma v kriptografske namene pa je bilo potrebno počakati do leta 1985. V tem letu je namreč ElGamal predstavil kriptosisteme z javnimi ključi, katerih varnost temelji na problemu diskretnega logaritma. Danes so

[4]Ueli M. Maurer in S. Wolf sta pokazala, da sta pri določenih predpostavkah DHP in DLP ekvivalentna problema [20].

poznani pod imenom ElGamalovi kriptosistemi [8]. Iz njihove modifikacije je nastala shema za digitalni podpis DSA (angl. Digital Signature Algorithm) [30, §7.4.2], ki je vključena v ameriški standard za digitalni podpis DSS (angl. Digital Signature Standard) [30, str. 286]. Danes je RSA najbolj uporaben in razširjen kriptosistem z javnimi ključi, kateremu pa so resna konkurenca tisti kriptosistemi, katerih varnost temelji na problemu diskretnega logaritma, in to predvsem zaradi DSS in hitrega razvoja kriptosistemov z eliptičnimi krivuljami, glej Menezes et al. [22, §8]. Pred ostalimi kriptosistemi z javnimi ključi je prednost kriptosistemov z eliptičnimi krivuljami v relativno kratkih dolžinah ključev za enak nivo varnosti in v občutno krajši programski kodi. Zato nam ostane za uporabne programe na voljo več prostora. V mnogih okoljih z omejenim prostorom so kriptosistemi z eliptičnimi krivuljami tudi edina možnost za kriptografijo z javnimi ključi, npr. na pametnih karticah. V razvoju kriptografije

simetrične šifre (AES, IDEA)	**asimetrične šifre** (RSA, DSA)	**asimetrične šifre** eliptične krivulje
40 bitov	274 bitov	80 bitov
56 bitov	384 bitov	106 bitov
64 bitov	512 bitov	132 bitov
80 bitov	**1024** bitov	**160** bitov
96 bitov	1536 bitov	185 bitov
112 bitov	2048 bitov	237 bitov
120 bitov	2560 bitov	256 bitov
128 bitov	**3072** bitov	**283** bitov
192 bitov	8192 bitov	409 bitov
256 bitov	16384 bitov	540 bitov

Tabela 1.1: Dolžine ključev z enakovredno varnostjo.

z javnimi ključi je bilo "razbitih" veliko predlaganih kriptosistemov. Obdržale so se le tri vrste, zato jih lahko danes smatramo za varne in učinkovite. Glede na matematični problem, na katerem je osnovana njihova varnost, jih delimo na kriptosisteme, ki temeljijo na naslednjih problemih:

- faktorizacija celih števil, npr. RSA kriptosistem, glej Stinson [30, §5.3] in Rabinov kriptosistem [30, §5.8];
- diskretni logaritem, npr. shema za digitalni podpis DSA [30, §7.4.2],

Diffie-Hellmanov protokol [7] in ElGamalovi kriptosistemi, glej Menezes et al. [23, §8.4 in §11.5];

- diskretni logaritem na eliptični krivulji, npr. shema za digitalni podpis ECDSA (angl. Elliptic Curve Digital Signature Algorithm) [30, §7.4.3].

Za nobenega od zgornjih kriptosistemov ni dokazov, da bi temeljili na resnično težkem matematičnem problemu. Lahko pa domnevamo, da so ti problemi izjemno težki, ker intenzivne študije s strani vodilnih matematičnih in računalniških strokovnjakov niso prinesle učinkovitih algoritmov za njihovo reševanje. Druga in tretja skupina kriptosistemov sta varianti ElGamalovih kriptosistemov, vendar pa je njuna varnost zasnovana na popolnoma različnih problemih diskretnega logaritma. Zaradi učinkovitejših algoritmov za reševanje matematičnih problemov v prvi in drugi skupini kriptosistemov je potrebno v kriptosistemih iz teh dveh skupin uporabljati občutno daljše ključe kot v kriptosistemih z eliptičnimi krivuljami. Na primer, dolžine ključev v kriptosistemih z eliptičnimi krivuljami se v današnji uporabi gibljejo od 160 do 256 bitov in nudijo enak nivo varnosti kot RSA in ostali kriptosistemi, ki so zgrajeni na problemu diskretnega logaritma z dolžinami ključev od 1024 do 2560 bitov, glej Jurišić in Menezes [11], Jurišić in Tonejc [15] in tabelo 1.1. V tej tabeli imamo napisane tudi dolžine ključev za enak nivo varnosti za simetrične šifre.

1.2 Varnost in diskretni logaritem

Naj bo G poljubna ciklična grupa reda n. Potem je izomorfna grupi $(\mathbb{Z}_n, +)$, glej Vidav [32, §2.9]. V grupi $(\mathbb{Z}_n, +)$ znamo učinkovito rešiti DLP z Evklidovim algoritmom. Če bi poznali kak izomorfizem iz grupe G v grupo $(\mathbb{Z}_n, +)$, potem bi lahko učinkovito rešili DLP v grupi G. Vse to bomo pokazali v prvem delu razdelka. V drugem delu bomo predstavili reševanje DLP v grupi $(\mathbb{Z}_n^*, *)$, kjer je število n produkt dveh različnih lihih praštevil, tj. n je tipična oblika RSA modula. Če bi imeli učinkovit algoritem za reševanje DLP v tej grupi, bi lahko izračunali dešifrirni eksponent v RSA kriptosistemu in s tem faktorizirali število n, glej Stinson [30, §5.7.2]. V zadnjem delu razdelka bomo predstavili reševanje DLP v grupi $(\mathbb{Z}_p^*, *)$, kjer je p praštevilo. Če bi imeli učinkovit algoritem za reševanje DLP v praštevilski podgrupi te grupe, bi lahko izračunali privatni

(tajni) ključ podpisovalca v algoritmu za digitalni podpis DSA ter tako ponarejali podpise. V zaključku razdelka poudarimo, da je DLP osrednjega pomena za vse pomembnejše kriptosisteme z javnimi ključi. Sedaj pa bomo definirali nekaj osnovnih pojmov.

Naj bo n naravno število in p praštevilo. V kriptografiji pogosto uporabljamo množico števil $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$ oziroma množico števil $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \,:\, D(a,n) = 1\}$. Na množici števil $\mathbb{Z}_n$ oziroma $\mathbb{Z}_n^*$ je namreč zgrajen RSA kriptosistem, kjer je $n = p\,q$ in sta p in q najmanj 384 ali bolje 512-bitni praštevili, tj. n je 768 ali 1024-bitno sestavljeno število. Množica $\mathbb{Z}_n^*$ je grupa za operacijo množenja po modulu n. V nadaljevanju bomo na kratko predstavili RSA kriptosistem, njegovo natančnejšo predstavitev pa si lahko ogledamo v Petkovem diplomskem delu [25, §3]. Naj bo a število iz množice $\mathbb{Z}_n$, ki predstavlja neko sporočilo. V RSA kriptosistemu šifriramo/odšifriramo sporočilo a s potenciranjem $a^b \bmod n$, kjer je b šifrirni/odšifrirni eksponent z lastnostjo $D(b, (p-1)(q-1)) = 1$. **Eulerjeva funkcija** $\varphi(n)$ je standardna oznaka za število celih števil na intervalu $[1, n]$, ki so tuja številu n. Z uporabo Eulerjeve funkcije in upoštevanjem definicije množice $\mathbb{Z}_n^*$ lahko zapišemo, da je $|\mathbb{Z}_n^*| = \varphi(n)$ in $|\mathbb{Z}_{\varphi(n)}^*| = \varphi(\varphi(n))$. Za RSA modul $n = p\,q$ je $\varphi(n) = (p-1)(q-1)$. V tem smislu je šifrirni/odšifrirni eksponent b v RSA kriptosistemu z modulom n iz množice števil $\mathbb{Z}_{\varphi(n)}^*$. V obsegu $\mathbb{F}_p$ pa je zgrajen DSA kriptosistem, glej Stinson [30, §7.4.2]. Z njim podpisujemo sporočila v 160-bitni praštevilski podgrupi grupe $(\mathbb{Z}_p^*, *)$, kjer je p najmanj 768 ali bolje 1024-bitno praštevilo, tj. p je 232 oziroma 309-mestno praštevilo.

DLP v $(\mathbb{Z}_n, +)$ in Evklidov algoritem

Problem diskretnega logaritma v grupi $(\mathbb{Z}_n, +)$ lahko predstavimo takole: za dana $g, h \in \mathbb{Z}_n$, kjer je g generator[5] grupe $(\mathbb{Z}_n, +)$, je potrebno rešiti naslednjo linearno kongruenco $x\,g \equiv h \pmod{n}$ oziroma linearno diofantsko enačbo $x\,g + k\,n = h$ za nek $k \in \mathbb{Z}$. Ker je g generator grupe $(\mathbb{Z}_n, +)$, je $D(g, n) = 1$ in zato je g hkrati tudi element grupe $(\mathbb{Z}_n^*, *)$. V tej grupi pa lahko učinkovito računamo inverzne elemente z uporabo razširjenega Evklidovega algoritma, glej Cohen [5, §1.3.2], katerega časovna zahtevnost je $O((\log_2 n)^2)$, primerjaj Jurišić [12, 4. domača naloga], Petek [25, §5.3] in Juvan [16]. Na primer, s števili g

[5] Vsako število $a \in \mathbb{Z}_n$ z lastnostjo $D(a, n) = 1$ je očitno generator grupe $(\mathbb{Z}_n, +)$.

in n nam razširjen Evklidov algoritem vrne taki števili s in t, da je $g\,s + n\,t = 1$. Potem je $g\,s \equiv 1 \pmod{n}$ in zato je $g^{-1} = s$ v grupi $(\mathbb{Z}_n^*, *)$. Rešitev linearne kongruence $x\,g \equiv h \pmod{n}$ lahko sedaj zapišemo z $x \equiv h\,s \pmod{n}$. Razširjen Evklidov algoritem je torej primer učinkovitega algoritma za reševanje DLP v grupi $(\mathbb{Z}_n, +)$.

Oglejmo si, kako bi lahko uporabili učinkovito reševanje DLP v grupi $(\mathbb{Z}_n, +)$ za reševanje DLP v poljubni ciklični grupi G reda n. Privzemimo, da poznamo nek izomorfizem ϕ iz grupe G v grupo $(\mathbb{Z}_n, +)$. Kot bomo videli na koncu tega podrazdelka, je to zelo močna predpostavka. Potem velja za vse $g_1, g_2 \in G$ in $x \in \mathbb{N}_0$

$$\phi(g_1\,g_2) = \phi(g_1) + \phi(g_2) \bmod n \quad \text{in} \quad \phi(g_1^x) = x\,\phi(g_1) \bmod n.$$

Naj bo g generator grupe G. Ker je $g^i \neq g^j$ za $i \not\equiv j \pmod{n}$, je $\phi(g^i) \not\equiv \phi(g^j) \pmod{n}$ za $i \not\equiv j \pmod{n}$ in zato je $\phi(g)$ generator grupe $(\mathbb{Z}_n, +)$. Naj bo $h \in G$ poljuben element, katerega diskretni logaritem v osnovi g iščemo. Reševanje danega DLP v grupi G lahko prenesemo na reševanje temu ustreznega DLP v grupi $(\mathbb{Z}_n, +)$ z uporabo izomorfizma ϕ, tj.

$$g^x = h \quad \Longleftrightarrow \quad x\,\phi(g) \equiv \phi(h) \pmod{n}.$$

Rešitev $x = \log_g h$ enačbe $g^x = h$ v grupi G dobimo z reševanjem enačbe $x\,\phi(g) \equiv \phi(h) \pmod{n}$ v grupi $(\mathbb{Z}_n, +)$, katere rešitev x izračunamo z uporabo razširjenega Evklidovega algoritma, tj.

$$x \equiv \phi(h)\,(\phi(g))^{-1} \pmod{n}.$$

Po drugi strani pomeni vsako reševanje DLP v ciklični grupi G reda n v bistvu tudi računanje nekega izomorfizma iz grupe G v grupo $(\mathbb{Z}_n, +)$. Naj bo ϕ ta izomorfizem, ki je definiran z $\phi(g^i) = i \bmod n$ za vsak $i \in \mathbb{N}_0$. Če je $h = g^i$ za neko število $0 \leq i < n$, je $\phi(h) = i$. Potem lahko pišemo: $\log_g h = \phi(h)$ za vsak $h \in G$. Sedaj bomo pokazali uporabo izomorfizma ϕ za reševanje DLP v poljubni grupi H, ki je izomorfna grupi G. Naj bo ψ izomorfizem iz grupe G v grupo H. To pomeni, da je element $\psi(g)$ generator grupe H. Preslikava

$$\mu : H \to \mathbb{Z}_n,$$

ki je definirana z $\mu(\psi(g)^i) = i \bmod n$ za vsak $i \in \mathbb{N}_0$, pa je izomorfizem iz grupe H v grupo $(\mathbb{Z}_n, +)$. Zato je $\mu(h) = \phi(\psi^{-1}(h))$ za vsak $h \in H$. Računanje izomorfizma μ je v bistvu reševanje DLP v grupi H. Zato pomeni izomorfizem ψ prenos reševanja DLP iz grupe G na njej izomorfno grupo H.

Zgoraj opisani način reševanja DLP si lahko najhitreje ponazorimo z reševanjem le-tega v grupi $(\mathbb{Z}_p^*, *)$, kjer je p praštevilo. Vsaka taka grupa je namreč ciklična, ima $p-1$ elementov in je zato očitno izomorfna grupi $(\mathbb{Z}_{p-1}, +)$. Zato bi lahko rešili DLP v grupi $(\mathbb{Z}_p^*, *)$ z uporabo nekega znanega izomorfizma ϕ iz grupe $(\mathbb{Z}_p^*, *)$ v grupo $(\mathbb{Z}_{p-1}, +)$. Za ilustracijo si oglejmo konkreten primer.

Primer. Naj bo npr. $p = 23$. Poiskali bomo diskretni logaritem elementa $h = 3$ pri osnovi $g = 2$ v grupi $(\mathbb{Z}_{23}^*, *)$. Število $2 \in \mathbb{Z}_{23}^*$ ni generator grupe $(\mathbb{Z}_{23}^*, *)$[6], saj je $2^{11} \equiv 1 \pmod{23}$. Zato pomeni računanje izbranega diskretnega logaritma v bistvu reševanje DLP v ciklični podgrupi te grupe, tj. v grupi $\langle 2 \rangle = \{1, 2, 3, 4, 6, 8, 9, 12, 13, 16, 18\}$, kjer smo $\langle g \rangle$ definirali v (1). Kljub temu ga lahko izračunamo po zgoraj opisani poti, tj. z uporabo nekega znanega izomorfizma iz grupe $(\mathbb{Z}_{23}^*, *)$ v grupo $(\mathbb{Z}_{22}, +)$. Kot že vemo, je grupa $(\mathbb{Z}_{23}^*, *)$ ciklična, ima 22 elementov, in je zato izomorfna grupi $(\mathbb{Z}_{22}, +)$. Če želimo definirati nek izomorfizem ϕ iz grupe $(\mathbb{Z}_{23}^*, *)$ v grupo $(\mathbb{Z}_{22}, +)$ s sliko nekega generatorja grupe $(\mathbb{Z}_{23}^*, *)$, moramo le-tega najprej izbrati. Na primer, ker je $D(5, 22) = 1$, je število 5 generator grupe $(\mathbb{Z}_{23}^*, *)$. Za vsako število $n \in \mathbb{N}$ pa je očitno 1 generator grupe $(\mathbb{Z}_n, +)$. Zato je zelo priročno, da definiramo izomorfizem $\phi : \mathbb{Z}_{23}^* \to \mathbb{Z}_{22}$ z $\phi(5) = 1$. Vendar je s tem izomorfizem ϕ podan v implicitni obliki. Kot bomo videli v nadaljevanju nam v splošnem samo poznavanje nekega implicitno podanega izomorfizma iz grupe $(\mathbb{Z}_p^*, *)$ v grupo $(\mathbb{Z}_{p-1}, +)$ nič ne pripomore k reševanju DLP v grupi $(\mathbb{Z}_p^*, *)$. Nadaljujmo z računanjem $\log_2 3$ v grupi $(\mathbb{Z}_{23}^*, *)$. Najprej prenesemo z uporabo izomorfizma ϕ reševanje enačbe $2^x \equiv 3 \pmod{23}$ v grupi $(\mathbb{Z}_{23}^*, *)$ na reševanje enačbe $x\,\phi(2) \equiv \phi(3) \pmod{22}$ v grupi $(\mathbb{Z}_{22}, +)$. Slednjo enačbo začnemo reševati tako, da najprej izračunamo števili $\phi(2)$ in $\phi(3)$. Računanje teh števil pa ne pomeni nič drugega kot reševanje ustreznih DLP v grupi $(\mathbb{Z}_{23}^*, *)$. Pri danem izomorfizmu ϕ je namreč $\phi(2) = \log_5 2$ in $\phi(3) = \log_5 3$. Ker delamo v majhni grupi, lahko dis-

[6] Število $g \in \mathbb{Z}_n^*$ je generator grupe $(\mathbb{Z}_n^*, *)$ natanko tedaj, ko je $g^{\varphi(n)/p} \not\equiv 1 \pmod{n}$ za vsako praštevilo p, ki deli $\varphi(n)$, glej Menezes et al. [23, dejstvo 2.132(iv)].

kretna logaritma $\phi(2)$ in $\phi(3)$ hitro izračunamo, tj. $\phi(2) = \phi(5^2) = 2\,\phi(5) \equiv 2 \pmod{22}$ in $\phi(3) = \phi(5^{16}) = 16\,\phi(5) \equiv 16 \pmod{22}$. Nato dobimo enačbo $2\,x \equiv 16 \pmod{22}$. Slednja enačba ima na intervalu $[0, 21]$ rešitvi: $x \equiv 8 \pmod{22}$ in $x \equiv 19 \pmod{22}$. Ker iščemo tako najmanjše število x, ki ustreza enačbi $2^x \equiv 3 \pmod{23}$, je $x \equiv 8 \pmod{22}$ iskana rešitev. Torej, $\log_2 3 = 8$ v grupi $(\mathbb{Z}_{23}^*, *)$. □

Edini problem takega načina reševanja DLP je ta, da nimamo učinkovite metode za računanje kakega izomorfizma ϕ iz grupe $(\mathbb{Z}_p^*, *)$ v grupo $(\mathbb{Z}_{p-1}, +)$. Vedno lahko sicer definiramo tak izomorfizem ϕ, ki slika nek generator g grupe $(\mathbb{Z}_p^*, *)$ v 1, tj. v generator grupe $(\mathbb{Z}_{p-1}, +)$. Vendar, če želimo izračunati diskretni logaritem poljubnega elementa h iz grupe $(\mathbb{Z}_p^*, *)$ v osnovi g, moramo izračunati $\phi(h)$, saj je v takem primeru $\log_g h = \phi(h)$, tj. imeti moramo polinomski algoritem za računanje vrednosti izomorfizma ϕ na poljubnem elementu iz grupe $(\mathbb{Z}_p^*, *)$. Torej, tudi če vemo, da sta grupi izomorfni, še ne pomeni, da poznamo učinkovit algoritem za ekspliciten opis kakega izomorfizma.

DLP v $(\mathbb{Z}_n^*, *)$ in RSA

Naj bo $n = p\,q$, kjer sta p in q različni lihi praštevili, e nek šifrirni eksponent (javni ključ) in d pripadajoč dešifrirni eksponent (tajni ključ) v RSA kriptosistemu. Število n je tipična oblika RSA modula. Najbolj pogost in najbolj očiten "napad" na RSA kriptosistem je faktorizacija števila n, glej Stinson [30, §5.6]. Ali si lahko pri tem pomagamo tudi z algoritmi za reševanje DLP v grupi $(\mathbb{Z}_n^*, *)$? Pritrdilen odgovor na to vprašanje bomo pokazali v nadaljevanju. Najprej si pa bomo ogledali poenostavljen scenarij, ki ga imamo pri podpisovanju z uporabo RSA kriptosistema, glej Stinson [30, str. 276]. Nato bomo pokazali, da pomeni iskanje dešifrirnega eksponenta d v RSA kriptosistemu v bistvu reševanje določenega primera DLP v grupi $(\mathbb{Z}_n^*, *)$.

Naj bo m element iz grupe $(\mathbb{Z}_n^*, *)$, ki predstavlja neko sporočilo in $c \in \mathbb{Z}_n$ nek podpis sporočila m. Sporočilo m podpiše podpisovalec s svojim tajnim ključem d, tj. izračuna $c = m^d \bmod n$. Recimo, da pridobimo ta podpis c za sporočilo m. V nadaljevanju bomo poskušali izračunati podpisnikov tajni ključ d z reševanjem enačbe $c = m^d \bmod n$. Najprej pa pokažimo, da je podpis c element grupe $(\mathbb{Z}_n^*, *)$, tj. da je $D(c, n) = 1$ oziroma $D(m^d \bmod n, n) = 1$. O

tem se lahko hitro prepričamo z uporabo naslednjih dejstev, ki veljajo za vse $i \in \mathbb{N}_0$:

(i) $D(m,n) = 1 \Rightarrow D(m^i, n) = 1$,

(ii) za $m^i < n$ je $m^i \bmod n = m^i$ in zato je $D(m^i \bmod n, n) = D(m^i, n)$,

(iii) za $m^i > n$ je $D(m^i \bmod n, n) = D(m^i, n)$.

Dejstvi (i) in (ii) sta očitni, dejstvo (iii) pa temelji na znani lastnosti naravnih števil, katera je tudi osnova Evklidovega algoritma.

Trditev 1.1 *Naj bosta a in b poljubni naravni števili, $a > b$. Potem je*

$$D(a,b) = D(a \bmod b, b).$$

Dokaz. Naj bo $d = D(a,b)$. Potem $d|a$ in $d|b$. Ker je $a \bmod b = a - (a \operatorname{div} b)\, b$, velja tudi $d|(a \bmod b)$. Naj bo c poljubno naravno število, ki deli b in $a \bmod b$. Potem c deli $a \bmod b + (a \operatorname{div} b)\, b = a$. Po definiciji največjega skupnega delitelja sledi, da c deli $d = D(a,b)$ in zato je $d = D(a \bmod b, b)$. ∎

Poznamo število n, sporočilo $m \in \mathbb{Z}_n^*$ in nek podpis $c \in \mathbb{Z}_n^*$ sporočila m, ne poznamo pa podpisnikovega tajnega števila d. Le-to je rešitev enačbe $c = m^d \bmod n$ v grupi $(\mathbb{Z}_n^*, *)$ in je hkrati tudi element grupe $(\mathbb{Z}_{\varphi(n)}^*, *)$. Utemeljimo slednje. Eksponent b v potenci $a^b \bmod n$, kjer je $a \in \mathbb{Z}_n^*$, računamo po modulu reda grupe $(\mathbb{Z}_n^*, *)$, ki je enak $\varphi(n)$, tj. $a^b \equiv a^{b \bmod \varphi(n)} \pmod n$ za vsako celo število b. To bi pomenilo, da je število d iz množice $\mathbb{Z}_{\varphi(n)}$. Ker pa v RSA kriptosistemu med vsakim šifrirnim eksponentom e in pripadajočim dešifrirnim eksponentom d velja relacija $e\,d \equiv 1 \pmod{\varphi(n)}$, je $D(d, \varphi(n)) = 1$. Zato je $d \in \mathbb{Z}_{\varphi(n)}^*$.

Ena rešitev enačbe $c = m^d \bmod n$ je podpisnikov tajni ključ d, vse ostale rešitve te enačbe v množici $\mathbb{N}$ pa so $d + k\,\varphi(n)$ za $k \in \mathbb{N}$. Ker je d najmanjše tako naravno število, za katerega velja enačba $c = m^d \bmod n$ v grupi $(\mathbb{Z}_n^*, *)$, je to število tudi diskretni logaritem elementa c pri osnovi m v grupi $(\mathbb{Z}_n^*, *)$. Vendar bi morala biti po definiciji diskretnega logaritma njegova osnova, tj. število m, generator grupe $(\mathbb{Z}_n^*, *)$. To pa se ne more nikoli zgoditi, ker ta grupa ni ciklična[7]. Kljub temu si lahko pri računanju števila d pomagamo z algoritmi za reševanje DLP v grupi $(\mathbb{Z}_n^*, *)$. Razmislek za to je naslednji. V

[7] Grupa $(\mathbb{Z}_n^*, *)$ je ciklična natanko tedaj, ko je $n = 2, 4, p^k$ ali $2\,p^k$, kjer je p liho praštevilo in $k \geq 1$, glej Menezes et al. [23, dejstvo 2.132(i)]

RSA kriptosistemu sta p in q neznani lihi praštevili. Vemo pa, da sta oblike $p = 2\,p' + 1$ in $q = 2\,q' + 1$, kjer sta p' in q' različni in prav tako neznani lihi praštevili. Po Lagrangeovem izreku deli red vsakega elementa grupe red grupe, glej Stinson [30, izrek 5.4]. Vsi mogoči redi elementov v grupi $(\mathbb{Z}_n^*, *)$ so delitelji njenega reda, ki je $\varphi(n) = (p-1)(q-1) = 2^2\,p'\,q'$. Izmed deliteljev števila $\varphi(n)$ poznamo samo 2 in 4, ostalih pa ne. Vemo pa, da so vsa manjša od n, ker je $\varphi(n) < n$. Zato lahko hitro določimo tako število m, da je njegov red veliko število. Če je npr. $m^2 \not\equiv 1 \pmod{n}$ in hkrati tudi $m^4 \not\equiv 1 \pmod{n}$, potem je red števila m veliko število. Tak m je potem generator ciklične podgrupe grupe $(\mathbb{Z}_n^*, *)$, katere red je veliko število. Sedaj vemo, da lahko hitro določimo število m z velikim redom v grupi $(\mathbb{Z}_n^*, *)$. Zato se vrnimo k iskanju dešifrirnega eksponenta d v RSA kriptosistemu. Najprej bomo pokazali, kako ga lahko izračunamo, če poznamo praštevili p in q, in nato še, kako ga izračunamo, če le-teh praštevil ne poznamo. To je tudi najbolj pogosto.

Recimo, da poznamo praštevili p in q oziroma p' in q'. Potem je $\varphi(n) = 2^2\,p'\,q'$. Zato lahko hitro izračunamo d iz e z uporabo Evklidovega algoritma. Če nismo lastniki RSA modula n, potem praštevil p in q oziroma p' in q' ne poznamo. Zato si bomo v nadaljevanju ogledali, ali lahko tudi v takem primeru izračunamo podpisnikov tajni ključ d.

V RSA kriptosistemu pozna praštevili p in q oziroma p' in q' samo podpisovalec, tj. lastnik dešifrirnega ključa d. Zato ne moremo izračunati dešifrirnega eksponenta d po zgoraj opisani poti. Kljub temu si lahko pri reševanju enačbe $c = m^d \bmod n$ pomagamo z metodo veliki-mali korak. Ker je red vsakega elementa m v grupi $(\mathbb{Z}_n^*, *)$ navzgor omejen z n, lahko vzamemo za dolžino koraka v metodi veliki-mali korak število $\lceil \sqrt{n} \rceil$. Če izberemo za RSA modul n vsaj 160-bitno število, potem postane uporaba metode veliki-mali korak nemogoča. Njeni ogromni prostorski zahtevnosti bi se lahko sicer izognili z uporabo Pollardove ρ-metode (glej §3) ali natančneje z uporabo Pollard-Floydove ρ-metode (glej §4). Vendar Pollardove ρ-metode ne moremo uporabiti pri reševanju enačbe $c = m^d \bmod n$, kjer je d neznano število, ker ne poznamo reda grupe $(\mathbb{Z}_n^*, *)$. V tem primeru je red grupe nujno potreben za določitev števila d, glej linearno kongruenco (3.9) na strani 60. Kot smo že omenili, je rešitev d enačbe $c = m^d \bmod n$ enolično določena po modulu $\varphi(n)$, ki je tudi red grupe $(\mathbb{Z}_n^*, *)$.

Podobno bi bilo poznavanje števila $\varphi(n)$ nujno potrebno tudi pri uporabi metode index-calculus za reševanje DLP v grupi $(\mathbb{Z}_n^*, *)$ (glej §2.2). Zato metode index calculus ne moremo uporabiti za reševanje DLP v grupi $(\mathbb{Z}_n^*, *)$. Ker pa je DLP v grupi $(\mathbb{Z}_n^*, *)$ vsaj tako zahteven kot je zahtevna faktorizacija števila n, moramo za RSA modul n izbrati vsaj 1024-bitno število, da preprečimo uporabo algoritmov za faktorizacijo, glej Menezes et al. [23, §3.8] in Menezes et al. [22, §6.9].

Povzemimo naše ugotovitve v kratek zaključek o varnosti RSA kriptosistema. V njem je priporočljivo za p in q izbrati 384 ali bolje 512-bitni praštevili. Potem RSA modul n predstavlja 768 oziroma 1024-bitno sestavljeno število. Takih števil še ne moremo razstaviti z danes znanimi algoritmi za faktorizacijo, glej Stinson [30, §5.6.4]. Reševanje DLP v takih grupah $(\mathbb{Z}_n^*, *)$ pa je vsaj tako zahtevno kot je zahtevna faktorizacija števila n. Zato določajo velikost varnega števila n v kriptosistemu RSA algoritmi za faktorizacijo in ne algoritmi za DLP. Zato se tudi domneva, da bi RSA prej "padel" na faktorizaciji kot na problemu diskretnega logaritma.

DLP v $(\mathbb{Z}_p^*, *)$ in DSA

Pri algoritmu za digitalni podpis DSA je nekoliko drugače. Njegova varnost res temelji na težavnosti reševanja DLP v grupi $(\mathbb{Z}_p^*, *)$, kjer je p praštevilo, in v njeni podgrupi praštevilskega reda, glej Menezes et al. [23, opomba 11.58]. Za reševanje DLP v grupi $(\mathbb{Z}_p^*, *)$ lahko uporabimo metodo index calculus, ki je podeksponentne zahtevnosti tako časovne in prostorske (glej §2.2). To dejstvo se vsekakor upošteva v DSA kriptosistemu. Zato moramo za p izbrati najmanj 768 ali bolje 1024-bitno praštevilo, da preprečimo uporabo metode index calculus. Za reševanje DLP v praštevilski podgrupi grupe $(\mathbb{Z}_p^*, *)$ lahko uporabimo Pollardovo ρ-metodo. Da preprečimo njeno uporabo, je dovolj, da izberemo 160-bitno praštevilsko podgrupo 1024-bitne grupe $(\mathbb{Z}_p^*, *)$. Po drugi strani pa lahko rešujemo DLP v praštevilski podgrupi grupe $(\mathbb{Z}_p^*, *)$ tudi z uporabo metode index calculus, glej Menezes et al. [23, §3.6.6]. Z metodo index calculus rešujemo DLP v praštevilski podgrupi grupe $(\mathbb{Z}_p^*, *)$ tako, da rešimo določena primera DLP v grupi $(\mathbb{Z}_p^*, *)$.

Naj bo H podgrupa reda q grupe $(\mathbb{Z}_p^*, *)$, kjer sta p in q taki praštevili, da je $p - 1 = q\,\ell$, kjer je ℓ neko naravno število. Naj bo g generator grupe

H in h njen poljuben element. Potem obstaja natanko določeno število $x = \log_g h \bmod q$, da je $h = g^x \bmod p$ v grupi H. Očitno je, da sta g in h hkrati tudi elementa grupe $(\mathbb{Z}_p^*, *)$. Naj bo α generator grupe $(\mathbb{Z}_p^*, *)$. Potem obstajata enolično določeni števili $y = \log_\alpha h \bmod (p-1)$ in $z = \log_\alpha g \bmod (p-1)$, da je $h = \alpha^y \bmod p$ in $g = \alpha^z \bmod p$. Potem lahko pišemo $x = \log_g h \bmod q = (\log_\alpha h)\,(\log_\alpha g)^{-1} \bmod q$. Ker sta h in g elementa praštevilskega reda q, je očitno $D(y, p-1) = \ell$ in $D(z, p-1) = \ell$. To pomeni, da sta števili y in z deljivi s številom ℓ. Sedaj izračunamo diskretni logaritem $x = \log_g h \bmod q$ takole

$$x = (y/\ell)\,(z/\ell)^{-1} \bmod q.$$

V kriptosistemu DSA preprečimo uporabo metode index calculus tako, da za praštevilo p izbremo vsaj 768 ali bolje 1024-bitno število. Potem postane namreč uporaba metode index calculus za reševanje DLP v grupi $(\mathbb{Z}_p^*, *)$ računsko neizvedljiva. To in pa seveda tudi možnost uporabe Pollardove ρ-metode se vsekakor upošteva v standardu za digitalni podpis DSS, kjer je navedeno tudi priporočilo, da se v kriptosistemu DSA uporablja 160-bitno praštevilsko podgrupo 1024-bitne grupe $(\mathbb{Z}_p^*, *)$.

Kot smo že omenili, podpisujemo z algoritmom DSA v 160-bitni praštevilski podgrupi grupe $(\mathbb{Z}_p^*, *)$, kjer je p 768 oziroma 1024-bitno praštevilo. Vendar je aritmetika, ki jo uporabljamo, iz obsega $\mathbb{F}_p$, tj. delamo s 768 oziroma 1024-bitnimi števili. S prav toliko velikimi števili računamo tudi v RSA kriptosistemu. Zato se nam sedaj pojavi vprašanje: ali obstajajo 160-bitne praštevilske podgrupe, v katerih bi lahko računali samo s prav toliko velikimi števili in ne da bi se pri tem zmanjšala težavnost računanja diskretnega logaritma v njih, tj. varnost diskretnega logaritma? Na srečo take grupe obstajajo. En razred takih grup si bomo ogledali v naslednjem razdelku.

1.3 Eliptične krivulje

V tem razdelku bomo definirali razred algebraičnih krivulj, ki jih imenujemo eliptične krivulje, glej Stinson [30, §6.5] in Vidav [33]. Eliptične krivulje nad končnimi obsegi so postale okrog leta 1985 zanimive za reševanje različnih kriptografskih problemov. Na primer, za faktorizacijo števil, glej Cohen [5, §10.3],

za ugotavljanje praštevilskosti [5, §9.2] in za gradnjo kriptosistemov z javnimi ključi, glej Jurišić in Menezes [11] in Menezes et al. [22, §8]. Prva, ki sta neodvisno predlagala uporabo eliptičnih krivulj v kriptografiji z javnimi ključi, sta bila V. Miller [24] in N. Koblitz [19] leta 1985. V ElGamalovih kriptosistemih sta zamenjala grupo $(\mathbb{Z}_p^*, *)$ z grupo na eliptični krivulji nad končnim obsegom, glej Menezes et al. [23, §8.4 in §11.5]. To je bil začetek razvoja tako imenovane "eliptične kriptografije", ki jo na kratko označimo z **ECC** (angl. Elliptic Curve Cryptosystem).

Začeli bomo z definicijo eliptične krivulje nad realnimi števili, ki jo označimo z $E(\mathbb{R})$ ali na kratko z E. Nato bomo definirali tako binarno operacijo na njej, da bo postala množica E skupaj s to operacijo grupa. Tej operaciji bomo priredili algebraične formule. Sledila bo predstavitev eliptične krivulje nad praštevilskim obsegom $\mathbb{F}_p$, kjer je $p > 3$, in nad obsegom $\mathbb{F}_{2^n}$, kjer je n naravno število. Kot bomo videli, veljajo za množico točk na eliptični krivulji nad končnim obsegom enake algebraične formule kot za eliptično krivuljo nad realnimi števili. Pri izpeljavi teh formul se upošteva le karakteristiko[8] obsega, nad katerim je definirana eliptična krivulja. Nato bomo predstavili problem diskretnega logaritma na eliptični krivulji nad končnim obsegom. Pripomnimo, da bomo uporabljali oznako E tako za eliptično krivuljo kot tudi za grupo na njej, saj gre za isto množico. Razdelek bomo zaključili s predstavitvijo strukture grupe na eliptični krivulji nad končnim obsegom.

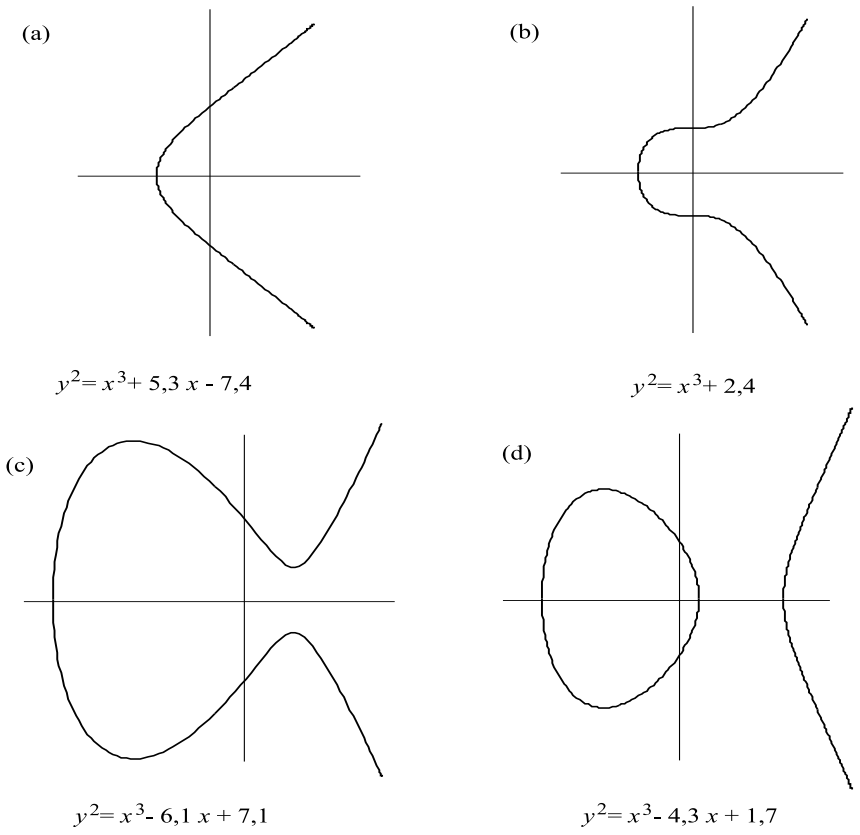
Eliptična krivulja nad realnimi števili

Naj bosta a in b taki realni števili, da je $4a^3 + 27b^2 \neq 0$. Za naše potrebe je **eliptična krivulja** množica vseh točk $(x, y) \in \mathbb{R} \times \mathbb{R}$, ki ustrezajo enačbi

$$y^2 = x^3 + a\,x + b, \tag{1.1}$$

skupaj s posebno točko $\mathcal{O}$, ki jo imenujemo **točka v neskončnosti**. Najbolj tipične oblike eliptične krivulje nad realnimi števili podamo na sliki 1.3. Pogoj $4a^3 + 27b^2 \neq 0$ je potreben in zadosten, da ima enačba $x^3 + a\,x + b = 0$ tri različne ničle, tj. tri realne ničle (npr. krivulja (d) na sliki 1.3) ali pa eno

[8] Karakteristika obsega K je najmanjše tako število $p \in K$, da je $p\,x = 0$ za vsak $x \in K$. Če je K neskončen obseg, je njegova karakteristika enaka 0, sicer je to neko praštevilo, glej Vidav [32, §3.6].



Slika 1.3: Najbolj tipične oblike eliptičnih krivulj nad realnimi števili.

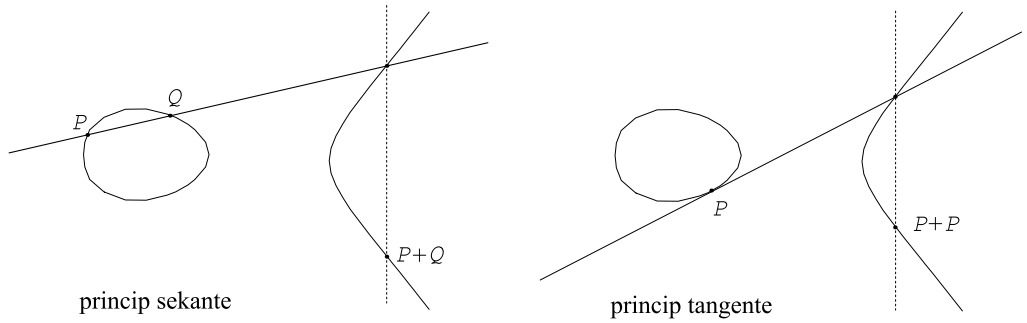
realno ničlo in dve konjugirano kompleksni ničli (npr. krivulje (a), (b) in (c) na sliki 1.3)[9]. Če pa je $4a^3 + 27b^2 = 0$, potem imenujemo krivuljo, podano z enačbo (1.1), **singularna eliptična krivulja**. Kot smo videli, je eliptična krivulja lahko singularna ali pa nesingularna. Nas bodo zanimale le nesingularne eliptične krivulje. Zato privzemimo od tu naprej, da so vse eliptične krivulje s katerimi delamo, nesingularne. Eliptična krivulja je sama po sebi zanimiv matematični objekt, vendar za kriptografske namene postane zanimiva takrat, ko nanjo uvedemo strukturo grupe.

Grupa na eliptični krivulji

Naj bo E poljubna eliptična krivulja nad realnimi števili. Sedaj bomo definirali seštevanje točk na množici E s pomočjo tako imenovanega **principa sekante**

[9]Naj bosta a in b poljubni kompleksni števili. Potem ima enačba $x^3 + a\,x + b = 0$ tri različne ničle natanko tedaj, ko je izraz $4a^3 + 27b^2$ različen od nič, glej Vidav [32, §6.9].

in tangente, glej sliko 1.4. To je geometrijsko pravilo, ki za dani točki iz



Slika 1.4: Geometrijski prikaz operacije seštevanja na eliptični krivulji nad $\mathbb{R}$.

množice E definira njuno vsoto. Naj bosta P in Q različni točki na E, ki nista $\mathcal{O}$. Skozi točki P in Q potegnemo premico. Če ni navpična, potem obstaja še tretje presečišče te premice z E. To je posledica tega, ker je enačba (1.1) za krivuljo E stopnje tri v spremenljivki x in ker definirana premica že seče krivuljo E v dveh točkah, glej trditev 1.2. Točko, ki jo dobimo s prezrcaljenjem tega tretjega presečišča preko abscisne osi, proglasimo za $P + Q$. Če je pa premica navpična, je v tem primeru smiselno definirati $P + Q = \mathcal{O}$. Če želimo točko P sešteti samo s seboj, tj. točko P podvojiti, uporabimo princip tangente. V točki P potegnemo tangento na krivuljo E. V tem primeru imamo v točki P dotikališče s krivuljo E. Če ta tangenta ni navpična, potem seka krivuljo E v natanko eni točki. To sledi iz tega, ker točko P upoštevamo kot dvojno skupno točko tangente in krivulje E in ker je enačba za krivuljo E stopnje tri v spremenljivki x, glej (1.1). Kot prej moramo to presečišče preslikati čez abscisno os. Tako dobljeno točko proglasimo za točko $P + P$ oziroma $2P$. Tudi tu obravnavamo primer, ko je tangenta navpična, posebej. V tem primeru je tudi smiselno definirati $P + P = \mathcal{O}$.

Trditev 1.2 *Naj bo K komutativen obseg, $p(x) \in K[x]$ poljuben polinom stopnje $n \geq 1$ in $a \in K$ neka njegova ničla, tj. $p(a) = 0$. Potem je*

$$p(x) = (x - a)\, q(x),$$

kjer je $q(x) \in K[x]$ nek polinom stopnje $n - 1$.

Dokaz. Naj bo $m \in \mathbb{N}$. Pri poljubnem $r \in K$ razstavimo polinom $x^m - r^m$ takole

$$x^m - r^m = (x - r)(x^{m-1} + r\,x^{m-2} + \cdots + r^{m-2}\,x + r^{m-1}) \tag{1.2}$$

Naj bo $p(x) = a_0 + a_1\,x + a_2\,x + \cdots + a_n\,x^n$, kjer $a_i \in K$ za $0 \leq i \leq n$ in $a_n \neq 0$. Ker je $p(a) = 0$, lahko pišemo

$$p(x) = p(x) - p(a) = a_1(x - a) + a_2(x^2 - a^2) + \cdots + a_n(x^n - a^n). \tag{1.3}$$

Vsak člen na desni strani (1.3) razstavimo po formuli (1.2), izpostavimo faktor $x - a$ in dobimo

$$p(x) = (x - a)\,q(x),$$

kjer je $q(x) = a_1 + a_2(x + a) + \cdots + a_n(x^{n-1} + a\,x^{n-2} + \cdots + a^{n-2}\,x + a^{n-1})$ polinom stopnje $n - 1$, ker je $a_n \neq 0$. ∎

S pomočjo elementarne geometrije lahko pretvorimo principa sekante in tangente v algebraično obliko. Na ta način dobimo eksplicitne algebraične formule za operacijo seštevanja točk na krivulji E. Kot bomo videli, so dobljene formule smiselne za poljuben komutativen obseg. Poglejmo si, kaj dobimo, če seštejemo poljubno točko $P = (x_1, y_1) \in E$, ki ni enaka $\mathcal{O}$, s točko $\mathcal{O}$. Vsaka premica skozi P in $\mathcal{O}$ je navpična. Krivuljo E seka še v točki

$$\overline{P} = (x_1, -y_1). \tag{1.4}$$

To sledi iz simetrije krivulje E glede na abscisno os, glej sliko 1.3 in (1.1). Vendar, ker je zrcalna slika točke $\overline{P}$ preko abscisne osi točka P, dobimo

$$P + \mathcal{O} = P.$$

Če je $P = \mathcal{O}$, potem je po definiciji $\mathcal{O} + \mathcal{O} = \mathcal{O}$. Zato je točka $\mathcal{O}$ nevtralni element oziroma enota za operacijo seštevanja. Ker je vsaka premica skozi točki P in $\overline{P}$ navpična, seka krivuljo E še v neskončni točki. Iz definicije seštevanja sledi

$$P + \overline{P} = \mathcal{O}. \tag{1.5}$$

Zato je točka $\overline{P}$ nasprotna točki P, tj. $-P = \overline{P}$. Nasprotna točka točke $\mathcal{O}$ pa je po definiciji kar ista točka, tj. $-\mathcal{O} = \mathcal{O}$. Naj bosta sedaj $P = (x_1, y_1)$ in

$Q = (x_2, y_2)$ poljubni točki iz množice E, ki nista enaki $\mathcal{O}$ in za kateri velja $P \neq -Q$. Točko $R = (x_3, y_3)$, za katero velja $R = P + Q$ v množici E, izračunamo po naslednjem pravilu:

$$x_3 = \lambda^2 - x_1 - x_2 \quad \text{in} \quad y_3 = \lambda(x_1 - x_3) - y_1, \tag{1.6}$$

kjer je

$$\lambda = \begin{cases} (y_2 - y_1)(x_2 - x_1)^{-1}, & \text{če} \quad P \neq Q \\ (3x_1^2 + a)(2y_1)^{-1}, & \text{če} \quad P = Q. \end{cases}$$

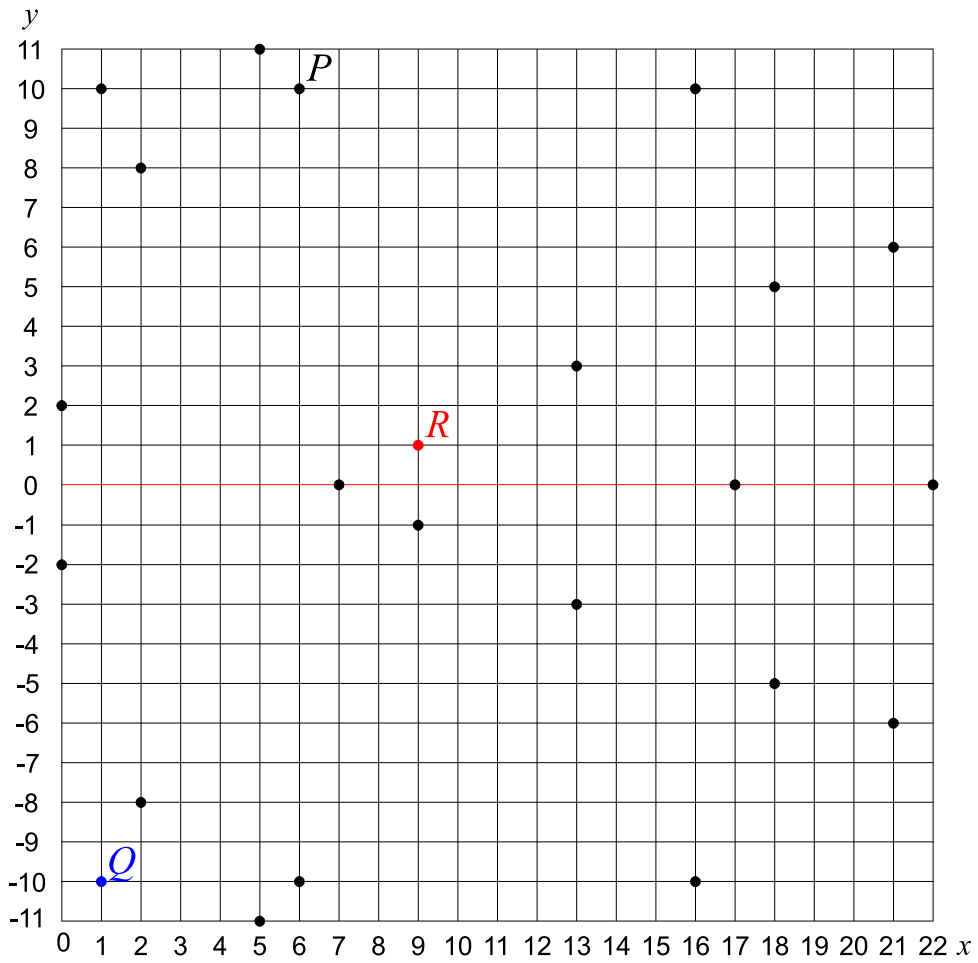
S precej dela lahko pokažemo, da je tako definirana operacija na množici E asociativna, tj. $(P + Q) + R = P + (Q + R)$ za vse $P, Q, R \in E$. Iz njene definicije lahko vidimo, da je tudi komutativna, tj. $P + Q = Q + P$ za vse $P, Q \in E$, glej sliko 1.4. Zato je množica E s to operacijo komutativna grupa, ki jo imenujemo **grupa na eliptični krivulji** in jo označimo z $\boldsymbol{(E, +)}$.

V kriptografiji uporabljamo predvsem končne matematične objekte. Zato bomo v nadaljevanju razdelka definirali eliptično krivuljo nad praštevilskim obsegom $\mathbb{F}_p$, kjer je $p > 3$, in nad obsegom $\mathbb{F}_{2^n}$, kjer je n naravno število. V splošnem jih lahko definiramo nad vsakim končnim obsegom, glej Menezes et al. [22, §7].

Eliptična krivulja nad praštevilskim obsegom

Naj bo $p > 3$ praštevilo. Eliptično krivuljo nad obsegom $\mathbb{F}_p$ definiramo na enak način kot smo jo definirali nad realnimi števili, glej (1.1). Razlika je le ta, da sta v tem primeru števili a in b iz obsega $\mathbb{F}_p$. Torej je eliptična krivulja nad obsegom $\mathbb{F}_p$ množica vseh tistih točk $(x, y) \in \mathbb{F}_p \times \mathbb{F}_p$, ki ustrezajo (1.1) skupaj s posebno točko $\mathcal{O}$. Na kratko označimo eliptično krivuljo nad praštevilskim obsegom $\mathbb{F}_p$ z $E_{a,b}(\mathbb{F}_p)$.

V množici $E_{a,b}(\mathbb{F}_p)$ definiramo operacijo seštevanja analogno kot v množici točk eliptične krivulje nad realnimi števili, glej sliko 1.4. Razlika je v tem, da tu izvajamo vse aritmetične operacije v obsegu $\mathbb{F}_p$. Točka $\mathcal{O}$ je nevtralni element za to operacijo seštevanja točk, tj. $P + \mathcal{O} = \mathcal{O} + P = P$ za vse $P \in E_{a,b}(\mathbb{F}_p)$. Za vsako točko $P = (x_1, y_1) \in E_{a,b}(\mathbb{F}_p)$ je točka $\overline{P} = (x_1, -y_1)$ njena nasprotna točka. Če je $(x, y) \in E_{a,b}(\mathbb{F}_p)$, potem je tudi $(x, -y) \in E_{a,b}(\mathbb{F}_p)$, glej (1.1). Zato je $P + \overline{P} = \mathcal{O}$ in pišemo $-P = \overline{P}$. Naj bosta $P = (x_1, y_1)$ in $Q = (x_2, y_2)$ poljubni točki iz množice $E_{a,b}(\mathbb{F}_p)$, ki nista enaki $\mathcal{O}$ in za kateri velja $P \neq -Q$.



Slika 1.5: Grafičen prikaz množice točk eliptične krivulje $y^2 = x^3 + 3\,x + 4$ nad obsegom $\mathbb{F}_{23}$. Za njene točke P, Q in R velja: $P + Q = R$.

Točko $R = (x_3, y_3)$, za katero velja $R = P+Q$ v množici $E_{a,b}(\mathbb{F}_p)$, pa izračunamo po pravilu (1.6). Z nekoliko več računskega dela lahko pokažemo, da je tako definirana operacija asociativna. Iz njene definicije pa je lahko videti, da je komutativna. Za kriptografijo je bistveno, da postane množica $E_{a,b}(\mathbb{F}_p)$ skupaj s to operacijo končna komutativna grupa, ki jo označimo z $(E_{a,b}(\mathbb{F}_p), +)$ ali na kratko z $E_{a,b}(\mathbb{F}_p)$.

Eliptično krivuljo $E_{a,b}(\mathbb{F}_p)$ lahko predstavimo geometrijsko tako, da označimo na mreži $\mathbb{F}_p \times \mathbb{F}_p$ tiste točke, ki ustrezajo dani enačbi krivulje. Tak način predstavitve podamo na sliki 1.5, ki je povzeta iz Certicomove domače strani [3]. Na tej sliki je tudi primer vsote točk dane eliptične krivulje. Eliptična krivulja

$E_{3,4}(\mathbb{F}_{23})$ ima 24 elementov. Kot vidimo na sliki 1.5, ležijo vse točke te krivulje simetrično glede na simetralo "$y = 0$". Na premici "$y = 0$" so vse ničle trinoma $x^3+3\,x+4$ nad obsegom $\mathbb{F}_{23}$. Simetrično ležeče točke glede na simetralo "$y = 0$" so si nasprotne, glej (1.4) in (1.5). Za točke na premici "$y = 0$" velja, da so same sebi nasprotne, saj je njihova y koordinata enaka 0, glej (1.4). Oglejmo si, kako izračunamo vsoto točk $P = (6, 10)$ in $Q = (1, -10) = (1, 13)$ na sliki 1.5 z uporabo pravila (1.6), kjer upoštevamo aritmetiko iz obsega $\mathbb{F}_{23}$.

$$\begin{aligned} P &= (6, 10) = (x_P, y_P) & \lambda &= (y_Q - y_P)\,(x_Q - x_P)^{-1} \bmod 23 \\ Q &= (1, 13) = (x_Q, y_Q) & &= 3 \cdot 18^{-1} \bmod 23 \\ P + Q &= (9, 1) = (x_R, y_R) = R & &= 3 \cdot 9 \equiv 4 \pmod{23} \end{aligned}$$

$$\begin{aligned} x_R &= \lambda^2 - x_P - x_Q \bmod 23 & y_R &= \lambda\,(x_P - x_R) - y_P \bmod 23 \\ &= 16 - 6 - 1 \bmod 23 & &= 4\,(6 - 9) - 10 \bmod 23 \\ &= 9 \bmod 23 & &= 4 \cdot 20 + 13 \equiv 1 \pmod{23} \end{aligned}$$

Eliptična krivulja nad obsegom s karakteristiko 2

Obsegi s karakteristiko 2 so najprimernejši za predstavitev njihovih elementov in njihove aritmetike v računalnikih, glej Jurišić in Menezes [11]. Zato imajo v kriptografiji z eliptičnimi krivuljami posebno mesto eliptične krivulje nad obsegi $\mathbb{F}_{2^n}$, kjer je n naravno število. Med njimi pa se najpogosteje uporabljajo tiste, ki so definirane z enačbo

$$y^2 + x\,y = x^3 + a\,x^2 + b, \tag{1.7}$$

kjer $a, b \in \mathbb{F}_{2^n}$ in $b \neq 0$. Eliptično krivuljo nad $\mathbb{F}_{2^n}$, ki je definirana z zgornjo enačbo (1.7) skupaj s točko $\mathcal{O}$, označimo z $E_{a,b}(\mathbb{F}_{2^n})$. Tudi tu je $\mathcal{O}$ dodatna točka eliptične krivulje, ki jo potrebujemo pri definiciji operacije seštevanja na množici $E_{a,b}(\mathbb{F}_{2^n})$. Le-to definiramo analogno kot nad realnimi števili, glej sliko 1.4. Pri izpeljavi za njo ustreznih algebraičnih formul upoštevamo aritmetiko iz obsega $\mathbb{F}_{2^n}$. Zaradi tega, ker delamo v obsegu s karakteristiko 2, so te algebraične formule nekoliko drugačne kot smo jih navedli do sedaj. Zato jih bomo v nadaljevanju eksplicitno napisali. Hkrati bomo navedli tudi lastnosti te operacije seštevanja. Točka $\mathcal{O}$ je nevtralni element za to operacijo

seštevanja točk, tj. $P + \mathcal{O} = \mathcal{O} + P = P$ za vse $P \in E_{a,b}(\mathbb{F}_{2^n})$. Za vsako točko $P = (x_1, y_1) \in E_{a,b}(\mathbb{F}_{2^n})$ je točka

$$\overline{P} = (x_1, x_1 + y_1) \tag{1.8}$$

njena nasprotna točka. Če je $(x, y) \in E_{a,b}(\mathbb{F}_{2^n})$, potem je tudi $(x, x + y) \in E_{a,b}(\mathbb{F}_{2^n})$, glej (1.7). Zato je $P + \overline{P} = \mathcal{O}$ in pišemo $-P = \overline{P}$. Naj bosta $P = (x_1, y_1)$ in $Q = (x_2, y_2)$ poljubni točki iz množice $E_{a,b}(\mathbb{F}_{2^n})$, ki nista enaki $\mathcal{O}$ in za kateri velja $P \neq -Q$. Točko $R = (x_3, y_3)$, za katero velja $R = P + Q$ v množici $E_{a,b}(\mathbb{F}_{2^n})$, izračunamo po naslednjem pravilu:

$$x_3 = \lambda^2 + \lambda + x_1 + x_2 + a \quad \text{in} \quad y_3 = (\lambda + 1)x_3 + \mu, \tag{1.9}$$

kjer je

$$\lambda = \begin{cases} (y_1 + y_2)(x_1 + x_2)^{-1}, & \text{če} \quad P \neq Q \\ x_1 + y_1\, x_1^{-1}, & \text{če} \quad P = Q \end{cases}$$

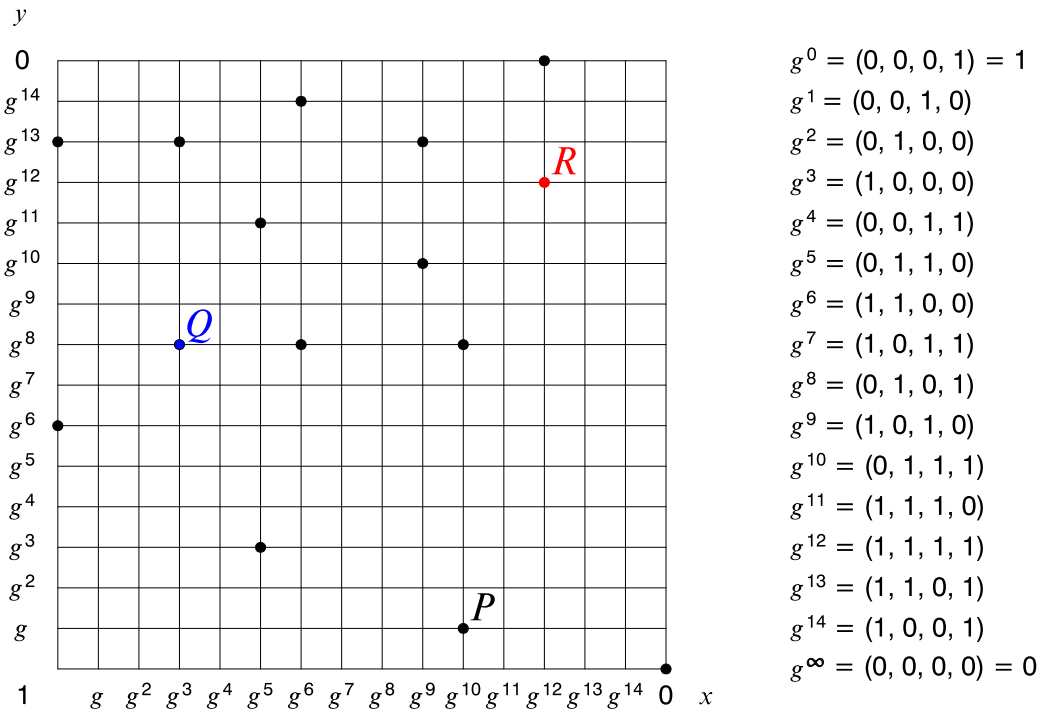
in

$$\mu = \begin{cases} (x_1\, y_2 + x_2\, y_1)(x_1 + x_2)^{-1}, & \text{če} \quad P \neq Q \\ x_1^2, & \text{če} \quad P = Q. \end{cases}$$

Z nekoliko več dela lahko pokažemo, da je tako definirana operacija asociativna. Iz njene definicije pa je lahko videti, da je komutativna, tj. $P + Q = Q + P$ za vse $P, Q \in \mathbb{F}_{a,b}(\mathbb{F}_{2^n})$. Za kriptografijo je bistveno, da je množica $E_{a,b}(\mathbb{F}_{2^n})$ skupaj s to operacijo končna komutativna grupa, ki jo označimo z $(E_{a,b}(\mathbb{F}_{2^n}), +)$ ali na kratko z $E_{a,b}(\mathbb{F}_{2^n})$. Oglejmo si primer eliptične krivulje nad obsegom $\mathbb{F}_{2^4}$, pred tem pa še postopek za konstrukcijo obsega $\mathbb{F}_{2^4}$.

Elemente obsega $\mathbb{F}_{2^4}$ lahko identificiramo z ekvivalenčnimi razredi faktorskega kolobarja $\mathbb{F}_2[x]/(f(x))$, kjer je $f(x) \in \mathbb{F}_2[x]$ nek nerazcepen polinom stopnje 4 nad obsegom $\mathbb{F}_2$. V tem smislu predstavlja množica polinomov nad obsegom $\mathbb{F}_2$, katerih stopnja je manjša od 4, elemente obsega $\mathbb{F}_{2^4}$. Torej $\mathbb{F}_{2^4} = \{a_3x^3 + a_2x^2 + a_1x + a_0 \,|\, a_i \in \{0, 1\}\}$. Vsak polinom $a_3x^3 + a_2x^2 + a_1x + a_0$ lahko zapišemo v vektorski obliki (a_3, a_2, a_1, a_0). Zato je $\mathbb{F}_{2^4} = \{(a_3, a_2, a_1, a_0) \,|\, a_i \in \{0, 1\}\}$. Aritmetiko v tako definiranem obsegu $\mathbb{F}_{2^4}$ izvajamo po modulu nerazcepnega polinoma $f(x)$. Množica neničelnih elementov obsega $\mathbb{F}_{2^4}$ je grupa za množenje po modulu nerazcepnega polinoma $f(x)$, glej Vidav [32, §9.5]. Označimo jo z $(\mathbb{F}_{2^4}^*, *)$. Le-ta je ciklična grupa reda $2^4 - 1$. Generator grupe

$(\mathbb{F}_{2^4}^*, *)$ imenujemo primitiven element oziroma generator obsega $\mathbb{F}_{2^4}$. Nerazcepen polinom $f(x) \in \mathbb{F}_2[x]$ stopnje m je **primitiven**, če je element $x = (0, \ldots, 0, 1, 0)$ generator grupe $(\mathbb{F}_{2^m}^*, *)$.



Slika 1.6: Grafičen prikaz množice točk eliptične krivulje $y^2 + x\,y = x^3 + g^4 x + 1$ nad obsegom $\mathbb{F}_{2^4}$. Za njene točke P, Q in R velja: $P + Q = R$.

V Menezes et al. [23, str. 158 in 159] lahko najdemo tabelo vseh tistih $1 \leq m \leq 1478$, za katere obstaja nerazcepen trinom nad obsegom $\mathbb{F}_2$ oblike $x^m + x^k + 1$, kjer je $1 \leq k \leq m - 1$. Za vsako tako število m je navedeno najmanjše tako število $1 \leq k \leq m-1$, da je trinom $x^m + x^k + 1$ nerazcepen nad obsegom $\mathbb{F}_2$. V [23, str. 161] pa lahko najdemo tabelo vseh tistih $1 \leq m \leq 229$, za katere obstaja primitiven polinom nad obsegom $\mathbb{F}_2$ oblike $x^m + x^k + 1$, kjer je $1 \leq k \leq m-1$. Tudi v tej tabeli je za vsako tako število m navedeno najmanjše tako število $1 \leq k \leq m - 1$, da je trinom $x^m + x^k + 1$ primitiven nad obsegom $\mathbb{F}_2$. V našem primeru definiramo obseg $\mathbb{F}_{2^4}$ s polinomom $f(x) = 1 + x + x^4$, ki je nerazcepen in primitiven nad $\mathbb{F}_2$,.

Napovedan primer eliptične krivulje nad obsegom $\mathbb{F}_{2^4}$ podamo na sliki 1.6, ki je povzeta iz Certicomove domače strani [3]. Na tej sliki imamo tudi vsoto

izbranih točk dane eliptične krivulje in vse elemente obsega $\mathbb{F}_{2^4}$, ki je generiran z elementom $g = x = (0, 0, 1, 0)$. Izberimo poljubno premico "$x = g^i$", kjer je $0 \leq i \leq 14$, na sliki 1.6. Če obstaja na izbrani premici kaka točka iz krivulje $E_{g^4,1}(\mathbb{F}_{2^4})$, potem je na njej tudi njena nasprotna točka, glej (1.8). Na premici "$x = 0$" pa so vse tiste točke dane eliptične krivulje $E_{g^4,1}(\mathbb{F}_{2^4})$, ki so same sebi nasprotne. Na slednji premici so namreč ničle polinoma $y^2 + 1 = 0$ nad obsegom $\mathbb{F}_{2^4}$, glej (1.7). Eliptična krivulja $E_{g^4,1}(\mathbb{F}_{2^4})$ ima 16 elementov. Vsoto točk $P = (g^{10}, g)$ in $Q = (g^3, g^8)$ na sliki 1.6 izračunamo z uporabo pravila (1.9). Preden začnemo s seštevanjem točk P in Q, si oglejmo, kako seštevamo v obsegu $\mathbb{F}_{2^4}$, ki je generiran z ničlo $g = x = (0, 0, 1, 0)$ polinoma $f(x) = 1 + x + x^4$. Elemente obsega $\mathbb{F}_{2^4}$ imamo podane v obliki potenc njegovega generatorja g. Zato hitro ugotovimo, da je množenje v tako predstavljenem obsegu lahko izvedljivo, tj. za poljubni celi števili i in j je $g^i\, g^j = g^{(i+j) \bmod 15}$. Po drugi strani pa seštevanje tako podanih elementov obsega $\mathbb{F}_{2^4}$ ni tako lahko. Za poljubni in različni celi števili i in j nimamo kakega enostavnega pravila po katerem bi lahko določili tako celo število k, da bi veljalo $g^i + g^j = g^k$. Vendar si lahko v primeru majhnega obsega, kot je to obseg $\mathbb{F}_{2^4}$, olajšamo operacijo seštevanja z uporabo tako imenovanih **Zechovih logaritmov**, glej Menezes et al. [22, §6.2]. Zechove logaritme moramo imeti seveda vnaprej izračunane, če si res želimo olajšati seštevanje. V primeru obsega $\mathbb{F}_{2^4}$ sestavimo tabelo Zechovih logaritmov (angl. Zech's log table) po naslednjem pravilu. Za vsako število $0 \leq i \leq 14$ moramo določiti tako število $j = z(i)$, da je $1 + g^i = g^{z(i)}$. Po dogovoru je $z(0) = \infty$, $z(\infty) = 0$ in $g^\infty = (0, 0, 0, 0)$. Zechov logaritem števila i imenujemo potem število $z(i)$. Zech log tabelo za seštevanje elementov obsega $\mathbb{F}_{2^4} = \mathbb{F}_2[x]/(1+x+x^4)$ podamo v tabeli 1.2. Pri računanju elementov tabele 1.2 si pomagamo s polinomom $f(x) = 1+x+x^4$. Na primer, hitro ugotovimo, da je $z(1) = 4$ in $z(4) = 1$, ker je $1+g = g^4$ oziroma $1 + g^4 = g$. Sedaj lahko za poljubni števili $0 \leq i, j \leq 14$ hitro določimo vsoto

$1 + g^i = g^{z(i)}$				
i	$z(i)$		i	$z(i)$
∞	0		7	9
0	∞		8	2
1	4		9	7
2	8		10	5
3	14		11	12
4	1		12	11
5	10		13	6
6	13		14	3

Tabela 1.2: Zech log tabela.

$g^i + g^j$ takole

$$g^i + g^j = g^i(1 + g^{(j-i) \bmod 15}) = g^i\, g^{z(j-i)} = g^{i+z(j-i)},$$

kjer eksponent $z(j-i)$ poiščemo v tabeli 1.2. Slednje pravilo uporabimo pri računanju $P + Q$ na sliki 1.6.

$$\begin{aligned} P &= (g^{10}, g) = (x_P, y_P) \\ Q &= (g^3, g^8) = (x_Q, y_Q) \end{aligned} \qquad \begin{aligned} \lambda &= (y_P + y_Q)\,(x_P + x_Q)^{-1} \\ &= (g + g^8)\,(g^{10} + g^3)^{-1} = g^{-2} = g^{13} \end{aligned}$$

$$\begin{aligned} P + Q &= R \\ R &= (g^{12}, g^{12}) = (x_R, y_R) \end{aligned} \qquad \begin{aligned} \mu &= (x_P\, y_Q + x_P\, y_P)\,(x_P + y_Q)^{-1} \\ &= (g^{10}\, g^8 + g^3\, g)\,(g^{10} + g^3)^{-1} \\ &= (g^3 + g^4)\, g^{-12} = g^{10} \end{aligned}$$

$$\begin{aligned} x_R &= \lambda^2 + \lambda + x_P + x_Q + g^4 \\ &= g^{13\cdot 2} + g^{13} + g^{10} + g^3 + g^4 = g^{12} \end{aligned} \qquad \begin{aligned} y_R &= (\lambda + 1)\, x_R + \mu \\ &= (g^{13} + 1)\, g^{12} + g^{10} = g^{12} \end{aligned}$$

DLP v grupi na eliptični krivulji nad končnim obsegom

Naj bo G končna multiplikativna grupa. V uvodu smo definirali DLP v ciklični podgrupi $\langle g \rangle \subseteq G$ reda n, glej (1). Sedaj si pa oglejmo definicijo problema diskretnega logaritma na eliptični krivulji E nad obsegom $\mathbb{F}_q$, kjer je $q = p^n$, p praštevilo in n naravno število. Označimo grupo na eliptični krivulji nad obsegom $\mathbb{F}_q$ z $(E(\mathbb{F}_q), +)$ ali na kratko z $E(\mathbb{F}_q)$. Naj bo E_1 njena ciklična podgrupa. Za dani točki $P, Q \in E_1$, kjer je P generator grupe E_1, definiramo DLP takole: poišči takšno število $x \in \{0, \ldots, |E_1| - 1\}$, da bo veljalo

$$Q = x\, P\, (= \overbrace{P + P + \cdots + P}^{x\text{-krat}}).$$

To je očitno problem diskretnega logaritma, ki ga pa imenujemo **problem diskretnega logaritma na eliptični krivulji** in označimo z **ECDLP** (angl. Elliptic Curve Discrete Logarithm Problem). ECDLP je v splošnem težak problem, tj. za njegovo reševanje ne poznamo polinomskega oziroma podeksponentnega algoritma, glej Stinson [30, §6.6]. Ker je grupa na eliptični krivulji komutativna, uporabljamo v kriptografiji z eliptičnimi krivuljami aditivno notacijo. Zaradi tega prihaja do zadrege pri poimenovanju diskretnega logaritma

v grupi na eliptični krivulji z istim imenom kot v multiplikativni grupi. Vendar kljub temu ločevanje med aditivnim in multiplikativnim zapisom ni pomembno, glej Jurišić in Menezes [11].

Struktura grupe na eliptični krivulji nad končnim obsegom

Množica točk eliptične krivulje E nad obsegom $\mathbb{F}_q$ je končna. Zaradi tega, da jo lahko varno uporabimo v kriptografiji, potrebujemo njeno natančno velikost. V začetku razvoja kriptografije z eliptičnimi krivuljami je bila že določitev števila točk na eliptični krivulji nad končnim obsegom velik problem. Od leta 1930 je bilo znano, da je to število omejeno na interval $[q+1-2\sqrt{q}, q+1+2\sqrt{q}]$. Ta rezultat imenujemo Hassejev izrek, glej Menezes et al. [22, izrek 7.7]. Vendar je matematik R. Schoof (izg. škof) kmalu po začetku razvoja kriptografije z eliptičnimi krivuljami iznašel polinomski algoritem za štetje točk na eliptični krivulji nad končnim obsegom. Danes je njegov algoritem poznan pod imenom Schoofov algoritem. Njegova časovna zahtevnost je $O((\log_2 q)^8)$, glej Stinson [30, str. 254]. Predstavitev Schoofovega algoritma imamo tudi v slovenski literaturi, glej Barbič [1, §4]. Iznajdba Schoofovega algoritma (leta 1985) je pomenila veliko prelomnico za uporabo eliptičnih krivulj v kriptografiji. Zadnje dosežke štetja točk na eliptični krivulji nad končnim obsegom najdemo na naslovu `(http://argote.ch/Records.html)`. Sedaj lahko izračunamo število točk na eliptični krivulji nad končnim obsegom z uporabo Schoofovega algoritma v polinomskem času. Za gradnjo kriptosistema z eliptičnimi krivuljami potrebujemo tako ciklično podgrupo grupe na eliptični krivulji, v kateri je problem diskretnega logaritma težak. Zato bi radi vedeli kaj več o strukturi grupe na eliptični krivulji. O tem nam govori naslednji izrek, kjer je $\cong$ oznaka za izomorfizem, glej Menezes et al. [22, izrek 7.10].

Izrek 1.3 *Naj bo E eliptična krivulja nad obsegom $\mathbb{F}_q$, kjer je $q = p^n$, p praštevilo in n naravno število. Potem je*

$$(E(\mathbb{F}_q), +) \cong (\mathbb{Z}_{n_1}, +) \oplus (\mathbb{Z}_{n_2}, +),$$

kjer $n_2 | n_1$ in $n_2 | (q-1)$. ∎

Vsaka grupa $(E(\mathbb{F}_q), +)$ je po zgornjem izreku ciklična, ali pa je izomorfna direktni vsoti cikličnih grup. Očitno je $n_2 = 1$ natanko tedaj, ko je $(E(\mathbb{F}_q), +)$

ciklična grupa. Če uspemo izračunati števili n_1 in n_2, potem vemo, da vsebuje $(E(\mathbb{F}_q), +)$ ciklično podgrupo. Slednja je osnova za gradnjo kriptosistema.

Poglavje 2

Računanje diskretnega logaritma

Učinovitega algoritma za reševanje DLP, ki bi ga lahko uporabili v poljubni grupi, nimamo. Zato je odločilnega pomena za varno uporabo problema diskretnega logaritma v kriptografske namene. Kljub temu pa je dobro poznati načine delovanja znanih algoritmov za reševanje DLP. To je tudi namen tega poglavja. V prvem razdelku je predstavljena metoda veliki-mali korak, v drugem pa metoda index calculus. V zadnjem razdelku je opisan Pohlig-Hellmanov algoritem.

2.1 Metoda veliki-mali korak

V tem razdelku bomo predstavili metodo veliki-mali korak za računanje diskretenga logaritma, glej Menezes et al. [23, §3.6.2]. Metodo veliki-mali korak imenujemo tudi Shanksova metoda velikega in malega koraka, glej Stinson [30, §6.2.1]. Le-ta je bila v začetku namenjena za računanje reda elementa končne grupe, glej Teske [31, str. 22]. Njena uporaba se je nato razširila tako na računanje diskretnega logaritma kot tudi na iskanje strukture končne komutativne grupe, glej Cohen [5, §5.4.1] ter na štetje točk na eliptični krivulji nad končnim obsegom, glej Barbič [1, §4.2.2].

Naj bo G ciklična grupa reda n, element $g \in G$ njen generator in $h \in G$ nek njen element, katerega diskretni logaritem v osnovi g iščemo. To pomeni, da obstaja natanko določeno število $x \in \{0, \ldots, n-1\}$, za katerega velja $h = g^x$. Naj bo $q = \lceil \sqrt{n} \rceil$. Število q imenujemo **dolžina koraka** v metodi veliki-mali

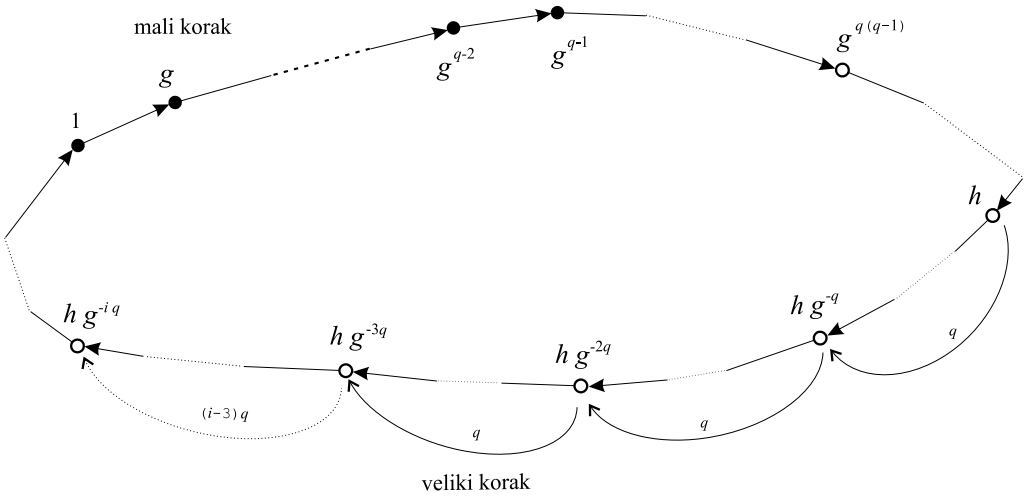
korak. Število x lahko enolično zapišemo v naslednji obliki

$$x = i\,q + j, \quad \text{kjer} \qquad i, j \in \mathbb{N}_0 \quad \text{in} \quad i, j < q.$$

Ker je $h = g^x = g^{i\,q} g^j$, je potem tudi $h(g^{-q})^i = g^j$. Taki števili i in j poiščemo v metodi veliki-mali korak tako, da najprej sestavimo tabelo **majhnih korakov**

$$R = \{(k, g^k) \ : \ 0 \le k \le q - 1\}.$$

Nato izračunamo element $\overline{g} = g^{-q}$. Z njegovo uporabo postopoma računamo elemente velikega koraka $h\,\overline{g}^{\,k}$ za $0 \le k < q$, dokler ne dobimo elementa iz tabele R. Za to je potrebno narediti največ q velikih korakov. Pri računanju elementov velikega koraka skačemo s korakom q, kar pomeni, da ne moremo preskočiti elementov malega koraka, glej sliko 2.1. Ko "trčimo" s tabelo malega koraka



Slika 2.1: Metoda veliki-mali korak.

R, poznamo i in j, ker je $h(g^{-q})^i = g^j$ ter lahko z njima izračunamo diskretni logaritem $x = i\,q + j$. Ko naredimo tabelo malih korakov, ima q elementov. To pomeni, da je prostorska zahtevnost metode veliki-mali korak $O(\sqrt{n}\,)$. Če je $x \ge q$, je časovna zahtevnost metode veliki-mali korak $q + \lfloor x/q \rfloor + O(\log q) = O(q) = O(\sqrt{n}\,)$ grupnih operacij. Pri tem je zahtevnost malega koraka q grupnih operacij, velikega koraka $\lfloor x/q \rfloor$ grupnih operacij in zahtevnost iskanja "trčenja" s tabelo malega koraka pa je $O(\log q)$. Za ilustracijo metode veliki-mali korak si oglejmo konkreten primer njene uporabe za reševanje DLP v grupi $(\mathbb{Z}_p^*, *)$, kjer

je p praštevilo. Pri tem upoštevamo, da računamo v množici števil $\mathbb{Z}_p$. Zaradi hitrejšega iskanja trčenja uredimo tabelo malega koraka po drugi komponenti naraščajoče.

Primer. Izračunali bomo diskretni logaritem elementa $h = 59$ pri osnovi $g = 3$ v grupi $(\mathbb{Z}^*_{137}, *)$. Ker je število 137 praštevilo, je $|\mathbb{Z}^*_{137}| = 136 = 2^3 \cdot 17$. Število 3 je generator grupe $(\mathbb{Z}^*_{137}, *)$, ker je $3^{68} \equiv 136 \pmod{137}$ in $3^8 \equiv 122 \pmod{137}$, zato je njegov red 136, glej opombo 6 na strani 13.

1. Izračunamo dolžino koraka $q = \lceil \sqrt{136} \rceil = 12$.
2. Izračunamo tabelo malega koraka $(j, 3^j \bmod 137)$ za $0 \leq j < 12$:

j	0	1	2	3	4	5	6	7	8	9	10	11
$3^j \bmod 137$	1	3	9	27	81	106	44	132	122	92	2	6

   Izračunano tabelo uredimo po drugi komponenti (naraščajoče)

j	0	10	1	11	2	3	6	4	9	5	**8**	7
$3^j \bmod 137$	1	2	3	6	9	27	44	81	92	106	**122**	132

3. Izračunamo $3^{-1} \equiv 46 \pmod{137}$ in zato je $\overline{g} = 3^{-12} \equiv 46^{12} \equiv 99 \pmod{137}$. Priredimo $f_0 := h = 59$.
4. Računamo elemente velikega koraka $f_i = f_0\, \overline{g}^{\,i} = 59 \cdot 3^{-12i} \equiv 59 \cdot 99^i \pmod{137}$ za $i = 0, 1, 2, \ldots$, dokler ne izračunamo takega elementa, ki ga že imamo v tabeli malega koraka. V našem primeru moramo izračunati devet elementov velikega koraka, tj.

i	0	1	2	3	4	5	6	7	**8**
$f_i = 59 \cdot 99^i \bmod 137$	59	87	119	136	38	63	72	4	**122**

   Dobimo $59 \cdot 3^{-8 \cdot 12} \equiv 122 \equiv 3^8 \pmod{137}$. To pomeni, da je $59 \equiv 3^{8+8 \cdot 12} \equiv 3^{104} \pmod{137}$. Torej je $\mathbf{\log_3 59 = 104}$ v grupi $(\mathbb{Z}^*_{137}, *)$. □

Metoda veliki-mali korak in RSA

V §1.2 smo pokazali, na kakšen način je vključena težavnost računanja diskretnega logaritma v varnost RSA kriptosistema. Tu si bomo natančneje ogledali uporabo metode veliki-mali korak pri "napadu" na RSA kriptosistem. V njem imamo RSA modul n, šifrirni eksponent e (javni ključ) in pripadajoč dešifrirni eksponent d (tajni ključ). Pri njegovi uporabi za podpisovanje dobimo tipično obliko problema diskretnega logaritma. Če pridobimo podpis c za

neko sporočilo $m \in (\mathbb{Z}_n^*, *)$, potem predstavlja enačba $c = m^d \bmod n$, kjer je d neznanka (tajni ključ), DLP v grupi $(\mathbb{Z}_n^*, *)$. Pri njenem reševanju si lahko pomagamo z metodo veliki-mali korak. Naj bo $q = \lceil n \rceil$ dolžina koraka. Število d, ki je rešitev enačbe $c = m^d \bmod n$ v grupi $(\mathbb{Z}_n^*, *)$, lahko enolično zapišemo kot $d = i\,q + j$, kjer je $0 \leq i, j < q$. Vendar, če bi želeli taki števili i in j res izračunati, bi potrebovali tabelo s q elementi iz grupe $(\mathbb{Z}_n^*, *)$, kar pa je za 768 oziroma 1024-bitni RSA modul n neizvedljivo. Lahko pa sestavimo tako tabelo z elementi malega koraka, ki jo še lahko spravimo na en ali več računalniških diskov. Če izberemo npr. za dolžino koraka $q = 2^{40}$, potem potrebujemo 1000 računalniških diskov po 100 Gb spomina za tabelo z 2^{40} elementi iz grupe $(\mathbb{Z}_n^*, *)$, kjer je n 1024-bitni RSA modul. Tabela malega koraka je $R = \{(0, 1), (1, m \bmod n), (2, m^2 \bmod n), \ldots, (2^{40} - 1, m^{2^{40}-1} \bmod n)\}$. Nato izračunamo število $\overline{m} = m^{-2^{40}} \bmod n$. To število uporabimo pri računanju števil velikega koraka. Pri tem računamo števila $c\,\overline{m}^i \bmod n$ za $i = 1, 2, \ldots, 2^{984}$. Z velikim korakom ne moremo preskočiti tabele malega koraka, zato se prej ali slej zgodi trčenje, tj. $c\,\overline{m}^i \equiv m^j \pmod n$ za neka $1 \leq i \leq 2^{984}$ in $1 \leq j \leq 2^{40}$. Potem je $c = m^{i\,2^{40}+j} \bmod n$ in zato je $d = i\,2^{40} + j$. V najslabšem primeru bi morali izračunati vse elemente velikega koraka. Zato je tudi ta način iskanja tajnega ključa d neizvedljiv. V tem primeru zaradi ogromne časovne zahtevnosti.

2.2 Metoda index calculus

Doslej smo si ogledali metodo veliki-mali korak v §2.1 in v uvodu metodo grobe sile za reševanje DLP. Obe lahko uporabimo v poljubni končni ciklični grupi in sta eksponentne časovne zahtevnosti. Zato je uporaba metode grobe sile in metode veliki-mali korak v grupah, ki se danes uporabljajo v kriptografiji, računsko prezahtevna. V tem razdelku si bomo ogledali metodo **index calculus**. Ta metoda predstavlja razred algoritmov, ki je podeksponentne zahtevnosti tako časovne in prostorske, glej Stinson [30, §6.2.4] in McCurley [21].

Naj bo G ciklična grupa reda n, element $g \in G$ njen generator in hkrati tudi osnova diskretnega logaritma. Naj bo $S = \{p_1, \ldots, p_t\}$ takšna podmnožica množice elementov iz grupe G, da lahko "večino" elementov iz G zapišemo

kot produkt elementov iz množice S. Množico S imenujemo **faktorska baza**. Metodo index calculus predstavimo v dveh korakih.

Prvi korak

Naj bo n red ciklične grupe G in element $g \in G$ njen generator. V prvem koraku izračunamo diskretne logaritme elementov iz faktorske baze S. To naredimo tako, da naključno izberemo celo število $a \bmod n$ in poskušamo zapisati element g^a samo z elementi iz faktorske baze S, tj.

$$g^a = p_1^{e_1} \cdots p_t^{e_t}. \tag{2.1}$$

V primeru, da nam uspe najti razcep (2.1), dobimo z logaritmiranjem enakosti (2.1) linearno kongruenco

$$a \equiv \left(e_1 \log_g p_1 + \cdots + e_t \log_g p_t\right) \pmod{n}. \tag{2.2}$$

V (2.2) so neznanke diskretni logaritmi elementov iz faktorske baze S. S t-kratno ponovitvijo zgornjega postopka lahko dobimo sistem t linearnih enačb s t neznankami, ki pa ni nujno rešljiv. Zato je dobro, da najdemo več kot t enačb oblike (2.2). Recimo $t + 10$, glej Menezes et al. [23, str. 110] in Stinson [30, str. 237]. Ko najdemo rešitev tega sistema enačb, ki jo lahko izračunamo v polinomskem času s postopnim izločevanjem neznank, je prvi korak končan.

Drugi korak

V drugem koraku poskušamo izračunati diskretni logaritem nekega elementa iz grupe G. Recimo, da želimo za element $h \in G$ najti njegov diskretni logaritem v osnovi g. To naredimo tako, da naključno izberemo celo število $s \bmod n$ in poskušamo zapisati element $g^s h$ samo z elementi iz faktorske baze S, tj.

$$g^s \; h = p_1^{s_1} \cdots p_t^{s_t}. \tag{2.3}$$

Če najdemo takšen zapis, dobimo z njegovim logaritmiranjem eksplicitno formulo

$$x = \log_g h \equiv \left(s_1 \log_g p_1 + \cdots + s_t \log_g p_t - s\right) \pmod{n}, \tag{2.4}$$

ki je rešitev danega diskretnega logaritma. S tem smo zaključili opis principa delovanja metode index calculus v poljubni ciklični grupi G reda n.

Zgornji opis metode index calculus ni natančen, saj nam ne pove, kako naj izberemo "primerno" faktorsko bazo S in kako naj učinkovito generiramo enačbe (2.1) in (2.3). To je tudi razlog, zaradi katerega metode index calculus ne moremo uporabiti v poljubni končni ciklični grupi. Faktorska baza mora biti "dovolj majhna", tako da sistem enačb v prvem koraku ni "prevelik". Hkrati pa mora biti delež elementov grupe, ki jih lahko zapišemo samo z elementi iz faktorske baze, "dovolj velik", tako da število poskusov za generiranje enačb (2.1) in (2.3) ni "preveliko". V procesu implementacije metode index calculus moramo poskušati čimbolj izenačiti zahtevnost obeh korakov. Izbira faktorske baze ima namreč velik vpliv na časovno zahtevnost obeh korakov metode index calculus. Če je faktorska baza "prevelika", potem bomo veliko časa porabili za računanje prvega koraka in obratno, "majhna" faktorska baza pomeni hiter izračun prvega koraka in precej dela za drugi korak. Takšno lastnost algoritma označujemo z izrazom časovna in prostorska usklajenost (angl. time-memory tradeoff).

Zaradi zgoraj povedanega o metodi index calculus ta ni uporabna v poljubni končni ciklični grupi. Učinkovita pa je v grupah, ki imajo še kakšno dodatno lastnost. Trenutno poznamo primerne faktorske baze in načine generiranja enačb (2.1) in (2.3) za multiplikativne grupe končnih obsegov $\mathbb{F}_q$, kjer je $q = p^n$, p praštevilo in n naravno število, glej Menezes et al. [23, §3.6.5]. V končnih obsegih imamo "posebne" elemente, ki so dobri kandidati za faktorsko bazo, ter metodo deljenja s poskušanjem (angl. trial division) za generiranje enačb (2.1) in (2.3), glej Stinson [30, str. 182]. V obsegih $\mathbb{F}_p$ so kandidati za faktorsko bazo S praštevila, v obsegih $\mathbb{F}_{p^n}$ pa nerazcepni polinomi. Pri implementaciji metode index calculus v praštevilskem obsegu $\mathbb{F}_p$ imamo na voljo štiri prijeme. Lahko uporabimo linearno rešeto (angl. linear sieve), rešeto ostankov (angl. residue sieve), metodo Gaussovih števil ali pa številsko rešeto (angl. number field sieve). Za predstavitev prvih treh glej Coppersmith et al. [6], za četrto pa glej Gordon [10].

Ob sprejemljivih predpostavkah je časovna in prostorska zahtevnost prvega koraka metode index calculus v praštevilskem obsegu $\mathbb{F}_p$

$$O\left(e^{(1+o(1))\sqrt{\ln p \, \ln\ln p}}\right),$$

časovna zahtevnost računanja posameznega diskretnega logaritma v drugem koraku pa je

$$O\left(e^{(1/2+o(1))\sqrt{\ln p \, \ln\ln p}}\right),$$

glej Coppersmith et al. [6]. Za ilustracijo metode index calculus si oglejmo konkreten primer njene uporabe za reševanje DLP v praštevilskem obsegu $\mathbb{F}_p$.

Primer. Naj bo npr. $p = 1201$. Poiskali bomo diskretni logaritem elementa $h = 501$ pri osnovi $g = 26$ v grupi $G = (\mathbb{Z}_{1201}^*, *)$. Ker je $26^{1200/P} \not\equiv 1 \pmod{1201}$ za vsako praštevilo P, ki deli $\varphi(1201) = 1200 = 2^4 \cdot 3 \cdot 5^2$, je število 26 res generator grupe G. Zato je 1200 red števila 26 v grupi G in tudi velikost grupe G. Naj bo $n = 1200$.

1. Za faktorsko bazo S izberemo prva štiri praštevila, tj. $S = \{2, 3, 5, 7\}$.

2. Potem naključno izbiramo števila $a \bmod 1200$. Pri vsakem izboru preverimo, če lahko razcepimo število $26^a \bmod 1201$ samo s praštevili iz množice S. Po nekaj ponovitvah uspemo zbrati naslednjih pet relacij:

$$\begin{aligned}
26^{74} \bmod 1201 = 28 &= 2^2 \cdot 7, \quad 26^{20} \bmod 1201 = 294 = 2 \cdot 3 \cdot 7^2, \\
26^{30} \bmod 1201 = 15 &= 3 \cdot 5, \quad 26^{18} \bmod 1201 = 672 = 2^5 \cdot 3 \cdot 7, \\
26^{82} \bmod 1201 = 567 &= 3^4 \cdot 7.
\end{aligned}$$

   Nato dobimo z logaritmiranjem zgornjih relacij pri osnovi 26 sistem petih kongruenc po modulu 1200 s štirimi neznankami:

$$\begin{aligned}
2\log_{26} 2 + \log_{26} 7 &\equiv 74, \quad \log_{26} 2 + \log_{26} 3 + 2\log_{26} 7 \equiv 20 \\
\log_{26} 3 + \log_{26} 5 &\equiv 30, \quad 5\log_{26} 2 + \log_{26} 3 + \log_{26} 7 \equiv 18 \\
4\log_{26} 3 + \log_{26} 7 &\equiv 82.
\end{aligned}$$

3. Zgornji sistem petih enačb s štirimi neznankami rešimo z uporabo postopnega izločevanja neznank. Dobimo

$$\log_{26} 2 = 412, \ \log_{26} 3 = 1108, \ \log_{26} 5 = 122 \text{ in } \log_{26} 7 = 450.$$

4. Sedaj naključno izberemo število $s \bmod 1200$ in preverimo, če lahko razcepimo število $501 \cdot 26^s \bmod 1201$ samo s praštevili iz množice S. Po nekaj

poskusih uspemo ugotoviti, da lahko zapišemo število $501 \cdot 26^{96} \bmod 1201$ samo z elementi iz faktorske baze S. Dobimo namreč število

$$501 \cdot 26^{96} \bmod 1201 = 1050,$$

ki ga lahko zapišemo z $1050 = 2 \cdot 3 \cdot 5^2 \cdot 7$. Po logaritmiranju slednje enakosti dobimo iskano rešitev

$$x = \mathbf{\log_{26} 501} = \log_{26} 2 + \log_{26} 3 + 2\log_{26} 5 + \log_{26} 7 - 96 \equiv \mathbf{918} \pmod{1200}$$

v grupi $(\mathbb{Z}^*_{1201}, *)$. □

2.3 Pohlig-Hellmanov algoritem

V tem razdelku bomo predstavili algoritem za DLP, ki sta ga razvila Pohlig in Hellman leta 1978, glej Stinson [30, §6.2.3] in Menezesa et al. [23, §3.6.4]. S tem algoritmom prevedemo vsak primer DLP iz grupe, ki ni praštevilskega reda, na več "manjših" primerov DLP. Gre za to, da lahko naredimo prehod na manjše primere DLP, če poznamo faktorizacijo reda grupe, v kateri rešujemo DLP in če ta faktorizacija sploh obstaja. V primeru grupe na eliptični krivulji nad končnim obsegom, ki je v današnjih uporabah reda velikosti od 160 do 256-bitnih števil, faktorizacija takih števil ne pomeni "velikega" problema. Zato se uporabi Pohlig-Hellmanovega algoritma v kriptografiji z eliptičnimi krivuljami izognemo tako, da izbiramo take grupe na eliptičnih krivuljah, katerih velikost je veliko praštevilo, npr. 160-bitno praštevilo.

Naj bo G ciklična grupa reda n, element $g \in G$ njen generator in $h \in G$ nek njen element, katerega diskretni logaritem v osnovi g iščemo. To pomeni, da obstaja natanko določeno število $x \in \{0, \ldots, n-1\}$, za katerega velja $h = g^x$. Število $x = \log_g h$ je enolično določeno po modulu n. Naj bo $n = p_1^{e_1} \cdots p_r^{e_r}$ praštevilska faktorizacija, kjer so $e_i \geq 1$ in p_i paroma različna praštevila. Ker poznamo faktorizacijo števila n, bi lahko izračunali $x \bmod n$ tudi tako, da bi najprej določili $x \bmod p_i^{e_i}$ za vse $1 \leq i \leq r$ in nato bi uporabili kitajski izrek o ostankih in izračunali $x \bmod n$, glej izrek 2.1. Algoritem, ki nam določi vse $x_i = x \bmod p_i^{e_i}$ za $1 \leq i \leq r$, je Pohlig-Hellmanov algoritem, glej Algoritem 2.1.

Izrek 2.1 (**Kitajski izrek o ostankih**) *Če so števila $n_1, n_2, \ldots, n_r$ paroma tuja, potem ima sistem kongruenc $x \equiv x_1 \pmod{n_1}, \ldots, x \equiv x_r \pmod{n_r}$ enolično rešitev po modulu $n = n_1 n_2 \cdots n_r$. Rešitev izračunamo z*

$$x = \sum_{i=1}^{r} x_i \, N_i \, M_i \bmod n,$$

kjer je $N_i = n/n_i$ in $M_i = N_i^{-1} \bmod n_i$ za $1 \leq i \leq r$. ∎

`Vhod:`	generator g grupe G reda n in element $h \in G$.
`Izhod:`	diskretni logaritem $x = \log_g h$.
	1. Naj bo $n = \prod_{i=1}^{r} p_i^{e_i}$ faktorizacija števila n, kjer so $e_i \geq 1$ in p_i paroma različna praštevila. 2. Za vsak $i \in \{1, 2, \ldots, r\}$ izračunamo število x_i, ki ga zapišemo v številskem sistemu z osnovo p_i, tj. $x_i = \ell_0 + \ell_1 p_i + \cdots + \ell_{e_i-1} p_i^{e_i-1}$, kjer $0 \leq \ell_j < p_i$, po naslednjih dveh korakih. 2.1 Naj bo $\gamma := 1$ in $\ell_{-1} := 0$. Definiramo element $\overline{g} := g^{n/p_i}$, ki je reda p_i, in je osnova diskretnega logaritma, v katerem bomo računali koeficiente ℓ_j za $j = 0, \ldots, e_i - 1$. 2.2 Za vsak indeks $j = 0, \ldots, e_i - 1$ definiramo $\gamma := \gamma \, g^{\ell_{j-1} p_i^{j-1}}$ in $\overline{h} := (h \, \gamma^{-1})^{n/p_i^{j+1}}$. Po trditvah 2.2 in 2.3 je $\overline{h} \in \langle \overline{g} \rangle$. Če je p_i do 80-bitno praštevilo, lahko izračunamo diskretni logaritem elementa $\overline{h}$ pri osnovi $\overline{g}$ z uporabo Pollardove ρ-metode. Tako dobimo koeficient $\ell_j := \log_{\overline{g}} \overline{h}$. 3. Dobili smo sistem kongruenc $x \equiv x_i \pmod{p_i^{e_i}}$ za $i = 1, \ldots, r$, katerega rešitev $x = \log_g h$ dobimo z uporabo kitajskega izreka o ostankih.

Algoritem 2.1: Pohlig-Hellmanov algoritem.

Oglejmo si pravilnost računanja števil $x_i = x \bmod p_i^{e_i}$ za $1 \leq i \leq r$ v Algoritmu 2.1. Za vsako praštevilo p_i, ki deli n, in pripadajočo potenco e_i velja

$$n \equiv 0 \pmod{p_i^{e_i}} \quad \text{in} \quad n \not\equiv 0 \pmod{p_i^{e_i+1}}.$$

V nadaljevanju bomo pokazali, kako računamo števila $x_i = x \bmod p_i^{e_i}$ za $1 \leq i \leq r$. (Zaradi večje preglednosti bomo namesto p_i in e_i pisali samo p oziroma e, oznake x_i pa ne bomo spremenili.) Ideja je v tem, da zapišemo število x_i v številskem sistemu z osnovo p, tj.

$$x_i = (\ell_{e-1}\ell_{e-2}\ldots\ell_1\ell_0)_p = \sum_{j=0}^{e-1} \ell_j p^j,$$

kjer $0 \leq \ell_j \leq p-1$ za $0 \leq j \leq e-1$. Od tod dobimo

$$x = \ell_0 + \ell_1 p + \cdots + \ell_{e-1}p^{e-1} + s\,p^e, \tag{2.5}$$

kjer je s neko naravno število. Torej je

$$x = \ell_0 + K\,p,$$

kjer je K neko naravno število. Z uporabo slednjega lahko pokažemo, da izračunamo koeficient $\ell_0 \bmod p$ v (2.5) z reševanjem enačbe

$$h^{n/p} = g^{\ell_0\, n/p}. \tag{2.6}$$

Trditev 2.2 *Za vsako praštevilo p, ki deli n, velja identiteta* (2.6).

Dokaz. Upoštevamo $x = \log_g h$ in (2.5). Dobimo

$$h^{n/p} = (g^x)^{n/p} = (g^{\ell_0+\ell_1 p+\cdots+\ell_{e-1}p^{e-1}+s\,p^e})^{n/p} = (g^{\ell_0+K\,p})^{n/p},$$

kjer je $K = \ell_1 + \ell_2 p + \cdots + \ell_{e-1}p^{e-2} + s\,p^{e-1}$. Ker je $g^{k\,n} = 1$ za vsako naravno število k, lahko zaključimo

$$(g^{\ell_0+K\,p})^{n/p} = g^{\ell_0\, n/p}\, g^{K\,n} = g^{\ell_0\, n/p}.$$

■

Naj bo $\overline{h} := h^{n/p}$ in $\overline{g} := g^{n/p}$. Za vsako praštevilo p je enačba (2.6) določen DLP v ciklični podgrupi $\langle\overline{g}\rangle$ reda p, katere rešitev je koeficient $\ell_0 = \log_{\overline{g}} \overline{h} \bmod p$. Vsak tak DLP lahko rešimo z uporabo Pollardove ρ-metode, če je p do 80-bitno število. Če je $e = 1$, smo z reševanjem enačbe (2.6) izračunali število $x_i = x \bmod p$, ker je $x = \ell_0 + s\,p$ za neko naravno število s. Če pa je $e > 1$, moramo nadaljevati z računanjem koeficientov $\ell_1, \ldots, \ell_{e-1}$ po modulu p. Računanje

slednjih je zelo podobno računanju koeficienta ℓ_0. Definirajmo $h_0 := h$ in za $1 \leq j \leq e-1$ pa

$$h_j := h\, g^{-(\ell_0+\ell_1 p+\cdots+\ell_{j-1}p^{j-1})}. \tag{2.7}$$

Tokrat lahko pokažemo, da izračunamo koeficient $\ell_j \bmod p$ z reševanjem splošnejše enačbe

$$h_j^{n/p^{j+1}} = g^{\ell_j\, n/p}. \tag{2.8}$$

Trditev 2.3 *Za vsako praštevilo p, ki deli n, velja identiteta* (2.8).

Dokaz. Upoštevamo $x = \log_g h$ in (2.7). Dobimo

$$h_j^{n/p^{j+1}} = (g^{x-\ell_0-\ell_1 p-\cdots-\ell_{j-1}p^{j-1}})^{n/p^{j+1}}.$$

Nato uporabimo (2.5) in zgornji zapis poenostavimo v

$$h_j^{n/p^{j+1}} = (g^{\ell_j\, p^j+\cdots+\ell_{e-1}p^{e-1}+s\, p^e})^{n/p^{j+1}}.$$

Slednje lahko preuredimo v

$$h_j^{n/p^{j+1}} = (g^{\ell_j\, p^j+K_j\, p^{j+1}})^{n/p^{j+1}},$$

kjer je $K_j = \ell_{j+1} + \ell_{j+2}p + \cdots + \ell_{e-1}p^{e-j-2} + s\, p^{e-j-1}$. Ker je $g^{k\,n} = 1$ za vsako naravno število k, lahko zaključimo

$$h_j^{n/p^{j+1}} = g^{\ell_j\, n/p}\, g^{K_j\, n} = g^{\ell_j\, n/p}.$$

■

Kot smo že omenili, določa število e število vseh koeficientov $\ell_j \bmod p$, ki jih moramo izračunati, da lahko določimo število x_i. Naj bo $\overline{h} := h_j^{n/p^{j+1}}$ in $\overline{g} := g^{n/p}$. Za vsako praštevilo p in za vsak $1 \leq j \leq e-1$ je enačba (2.8) določen DLP v ciklični podgrupi $\langle\overline{g}\rangle$ reda p, katere rešitev je koeficient $\ell_j = \log_{\overline{g}} \overline{h} \bmod p$. Pri tem velja omeniti, da je za $j = 0$ enačba (2.8) enaka enačbi (2.6). Tudi tu lahko izračunamo koeficient $\ell_j \bmod p$ z reševanjem enačbe (2.8) z uporabo Pollardove ρ-metode, če je p do 80-bitno praštevilo. Ko enkrat poznamo koeficient $\ell_j \bmod p$, se lahko hitro prepričamo (z uporabo definicije (2.7)), da lahko izračunamo element h_{j+1} z uporabo elementa h_j s preprosto rekurzijo

$$h_{j+1} = h_j\, g^{-\ell_j\, p^j}. \tag{2.9}$$

To pomeni, da je postopek računanja števila $x_i = x \bmod p$ tak: najprej izračunamo koeficient $\ell_0 \bmod p$ z reševanjem enačbe (2.8), ki je za $j = 0$ enaka enačbi (2.6), nato izračunamo element h_1 z uporabo rekurzije (2.9). Sledi računanje koeficienta $\ell_1 \bmod p$ z reševanjem enačbe (2.8) in nato računanje elementa h_2 z uporabo rekurzije (2.9). To pomeni, da izmenično uporabljamo identiteti (2.8) in (2.9), dokler ne izračunamo elementa h_{e-1} z uporabo rekurzije (2.9) in koeficienta $\ell_{e-1} \bmod p$ z reševanjem enačbe (2.8).

V Pohlig-Hellmanovemu algoritmu potrebujemo faktorizacijo števila n. Če poznamo faktorizacijo števila n, tj. $n = p_1^{e_1} \cdots p_r^{e_r}$, kjer so $e_i \geq 1$ in p_i paroma različna praštevila, potem lahko reduciramo dan primer DLP na $e_1 + e_2 + \cdots + e_r \leq \log n$ primerov DLP, ki jih imenujemo "podproblemi" danega DLP. Če so praštevilski faktorji števila n "majhni", so podproblemi hitro rešljivi. Podprobleme danega DLP lahko rešimo npr. z uporabo Pollardove ρ-metode, če so p_i do 80-bitna praštevila.

Naj bo $k = \sum_{i=1}^{r} e_i$, tj. število k označuje vsoto vseh potenc praštevil v faktorizaciji števila n. Zahtevnost računanja elementov h_j je $O(\log n)$ grupnih operacij. Vsako število $x_i = x \bmod p_i^{e_i}$ določimo tako, da rešimo e_i primerov DLP v ciklični podgrupi reda p_i z uporabo Pollardove ρ-metode. Zato je pričakovana časovna zahtevnost slednje faze $O(\sqrt{p_i}\,)$ grupnih operacij. Sedaj lahko ocenimo pričakovano časovno zahtevnost Pohlig-Hellmanovega algoritma z

$$O\left(\sum_{i=1}^{r} e_i(\log n + \sqrt{p_i}\,)\right) \subseteq O\left(k(\log n + \sqrt{p}\,)\right) \subseteq O(\sqrt{p}\,),$$

kjer je p največje praštevilo, ki nastopa v faktorizaciji števila n. Prostorska zahtevnost Pohlig-Hellmanovega algoritma je konstantna, če uporabimo Pollard-Floydovo ρ-metodo za reševanje podproblemov danega DLP, glej §4.

Poglavje 3

Pollardova ρ-metoda

Kot je bilo omenjeno že v uvodu, je Pollardova ρ-metoda trenutno najboljši algoritem za reševanje DLP v grupi na eliptični krivulji nad končnim obsegom. Zato je cilj tega poglavja in hkrati tudi glavna naloga tega dela podroben opis te metode. Poglavje je razdeljeno takole: v prvem razdelku so definirani osnovni pojmi, v drugem pa Pollardova ρ-metoda. Njena prostorska in časovna zahtevnost je utemeljena v tretjem razdelku. V četrtem razdelku je predstavljena uporaba Pollardove ρ-metode za reševanje DLP. V zadnjem razdelku pa je predstavljena Pollardova ρ-metoda za faktorizacijo sestavljenega števila.

3.1 Periodičnost, trčenje in naključnost zaporedij

Naj bo G neprazna množica in $y_0, y_1, \ldots$ neko zaporedje njenih elementov, ki ga na kratko označimo z $\{y_i\}$. Če obstajata takšni števili $m > 0$ in $n_0 \geq 0$, da velja $y_{n+m} = y_n$ za vsak $n \geq n_0$, potem imenujemo tako zaporedje **(končno) periodično**. Število m imenujemo **perioda** zaporedja, število n_0 pa **predperioda** zaporedja. Najmanjšo od vseh period periodičnega zaporedja imenujemo **osnovna perioda** zaporedja. Če ima periodično zaporedje $\{y_i\}$ osnovno periodo m, potem imenujemo najmanjše nenegativno število n_0, za katerega velja $y_{n+m} = y_n$ za vsak $n \geq n_0$, **osnovna predperioda**.

Trditev 3.1 *Naj bo $\{y_i\}$ poljubno periodično zaporedje elementov iz neprazne množice G. Potem je vsaka perioda zaporedja $\{y_i\}$ deljiva z osnovno periodo.*

Dokaz. Naj bo m neka perioda končno periodičnega zaporedja $\{y_i\}$ in m_1 njegova osnovna perioda. To pomeni, da velja $y_{n+m} = y_n$ za vse $n \geq n_0$ in $y_{n+m_1} = y_n$ za vse $n \geq n_1$, kjer sta n_0 in n_1 ustrezni števili iz definicije končno periodičnega zaporedja. Po definiciji osnovne periode pa je $m_1 \leq m$. Privzemimo, da m ni deljiv z m_1. Potem je očitno $m_1 < m$. Zato lahko enolično zapišemo $m = k\,m_1 + r$, kjer je $k \geq 1$ in $0 < r < m_1$. To pomeni, da za vsa števila $n \geq \max(n_0, n_1)$ velja

$$y_n = y_{n+m} = y_{n+k\,m_1+r} = y_{n+(k-1)m_1+r} = \cdots = y_{n+r}.$$

Zato je r perioda zaporedja. To pa je v protislovju z definicijo osnovne periode. ■

Naj bosta i in j različni naravni števili. Če velja $y_i = y_j$ in $i \neq j$, kjer sta y_i in y_j elementa iz zaporedja $\{y_i\}$, potem pravimo, da je prišlo do **trčenja**. Eksistenca trčenja je očitna v vsakem periodičnem zaporedju. V tem delu se lahko omejimo samo na zaporedja z elementi iz neke končne in neprazne množice G. Kot bomo videli v trditvi 3.2, je vsako zaporedje $\{y_i\}$ v množici G, ki ga definiramo z rekurzijo $y_{i+1} = f(y_i)$ za $i \geq 0$, kjer je $f : G \to G$ neka funkcija in $y_0 \in G$ nek začetni element, periodično.

Trditev 3.2 *Naj bo G končna neprazna množica ter $f : G \to G$ neka funkcija. Potem je zaporedje $\{y_i\}$, definirano z neko začetno vrednostjo $y_0 \in G$ in rekurzivno enačbo*

$$y_{i+1} = f(y_i) \text{ za } i \geq 0, \tag{3.1}$$

periodično.

Dokaz. Po Dirichletovem principu se po $|G|+1$ korakih ponovi vsaj en element v zaporedju $\{y_i\}$. Zato obstajata taki pozitivni števili $m \geq 1$ in $n_0 \geq 0$, da velja $y_{n_0+m} = y_{n_0}$. Naj bosta m in n_0 taki števili, da je y_{n_0} prvi element v zaporedju $\{y_i\}$, ki se ponovi, in $n_0 + m$ najmanjši indeks, za katerega je $y_{n_0+m} = y_{n_0}$. Pokazali bomo, da je $y_{n'+m} = y_{n'}$ za vsak $n' \geq n_0$. Če je $n' = n_0$, je to očitno iz definicije m in n_0. Predpostavimo, da trditev velja za $n = n_1 \geq n_0$ in se prepričajmo, da je izpolnjena tudi za $n = n_1 + 1$. Hitro vidimo, da je

$$y_{n_1+1+m} = f(y_{n_1+m}) = f(y_{n_1}) = y_{n_1+1}.$$

Torej je $\{y_i\}$ res periodično zaporedje. ■

Naj bo zaporedje $\{y_i\}$ definirano v končni in neprazni množici G s pravilom (3.1). Osnovno predperiodo vsakega takega zaporedja $\{y_i\}$ bomo od sedaj naprej označevali z $\boldsymbol{\mu}$, njegovo osnovno periodo pa z $\boldsymbol{\lambda}$. Hkrati pa imenujemo $\{y_0, y_1, \ldots, y_\mu\}$ **rep** in $\{y_\mu, y_{\mu+1}, \ldots, y_{\mu+\lambda-1}\}$ **cikel** zaporedja $\{y_i\}$. Pripomnimo naslednje: zaporedje $\{y_i\}$ nima predperiode, tj. $\mu = 0$, natanko tedaj, ko je f injektivna funkcija, glej sliko 3.1 na strani 51. Na tej sliki lahko hitro vidimo, da bi v primeru, ko je $\mu > 0$ veljalo $f(y_{\mu-1}) = y_\mu$ kot tudi $f(y_{\mu+\lambda-1}) = y_\mu$, pri čemer sta $y_{\mu-1}$ in $y_{\mu+\lambda-1}$ različna elementa, vendar je to v protislovju z injektivnostjo funkcije f.

Primer. Naj bo G množica četveric števil $(x_i, x_{i+1}, x_{i+2}, x_{i+3})$, kjer je $i \geq 0$ in $x_i, x_{i+1}, x_{i+2}, x_{i+3} \in \mathbb{Z}_{10}$. Množica G ima očitno 10^4 elementov. Definirajmo funkcijo $f : G \to G$ takole

$$f((x_1, x_2, x_3, x_4)) = \sum_{i=1}^{4} x_i \bmod 10 \quad \text{za} \quad i \geq 0.$$

Naj bo $y_0 := (1, 9, 8, 1)$ začetni element zaporedja $\{y_i\}$ v množici G, ki ga definiramo rekurzivno z $y_{i+1} := f(y_i)$ za $i \geq 0$. Zaporedje $\{y_i\}$ lahko zapišemo tudi kot

$$\{x_i\} = \{x_0, x_1, \ldots\} = \{1, 9, 8, 1, 9, 7, 5, 2, 3, 7, 7, 9, 6, 1, 5, 1, \ldots\}.$$

Hitro se lahko prepričamo, da tako definirano zaporedje $\{y_i\}$ nima repa. O tem se prepričamo tako, da pokažemo, da je funkcija $f : G \to G$, ki rekurzivno definira to zaporedje, injektivna. Izberimo si neko četverico števil $(x_{j+1}, x_{j+2}, x_{j+3}, x_{j+4})$ v zaporedju $\{x_i\}$ in preverimo, če je število $x_j \in \{x_i\}$ res rešitev enačbe $x + x_{j+1} + x_{j+2} + x_{j+3} \equiv x_{j+4} \pmod{10}$, kjer je $j \geq 0$. Na primer, za četverico $(2, 3, 7, 7)$ dobimo enačbo $x + 2 + 3 + 7 \equiv 7 \pmod{10}$, katere rešitev je število 5. To število je v zaporedju $\{x_i\}$ res pred četverico $(2, 3, 7, 7)$.

Kateri element iz množice G se v zaporedju $\{y_i\}$ prvi ponovi? Kot smo se že prepričali, zaporedje $\{y_i\}$ nima repa. Zato je očitno, da je začetni element zaporedja $\{y_i\}$, tj. četverica $y_0 = (1, 9, 8, 1)$, tisti element iz množice G, ki se v zaporedju $\{y_i\}$ prvi ponovi. S kratkim računalniškim programom se lahko prepričamo, da je cikel tako definiranega zaporedja $\{y_i\}$ dolžine 1560. Če izberemo za začetno vrednost zaporedja $\{y_i\}$ kak drug element, se lahko zgodi,

da dobimo drugačno zaporedje v množici G. Za $y_0 = (0, 1, 1, 0)$ dobimo npr. zaporedje z dolžino cikla 1560, ki pa ima vse elemente različne od prej definiranega zaporedja $\{y_i\}$. V mnnožici G lahko najdemo 6 paroma disjunktnih zaporedij z dolžino cikla 1560, dve z dolžino cikla 312, tri z dolžino cikla 5 in eno zaporedje z enim elementom v ciklu. To so tudi vsi paroma disjunktni cikli v množici G, ki jih dobimo z uporabo zgoraj definirane funkcije f, saj je $6{\cdot}1560{+}2{\cdot}312{+}3{\cdot}5{+}1{\cdot}1 = 10\,000$. Lahko se prepričamo, da generirajo npr. elementi $(1, 9, 8, 1), (0, 1, 1, 0), (0, 0, 0, 1), (1, 2, 3, 4), (6, 1, 6, 6)$ in $(7, 7, 7, 7)$ paroma disjunktne cikle dolžine 1560, elementa $(0, 0, 2, 2)$ in $(2, 2, 2, 2)$ disjunktna cikla dolžine 312 in elementi $(0, 0, 5, 5), (0, 5, 0, 5)$ in $(5, 5, 5, 5)$ disjuntne cikle dolžine 5. Element $(0, 0, 0, 0)$ očitno predstavlja edini cikel dolžine 1. □

V splošnem se elementi, ki so na repu zaporedja $\{y_i\}$, ki je definirano rekurzivno z (3.1) in z neko začetno vrednostjo y_0, v njem nikoli več ne ponovijo. Še več, elementi $y_0, \ldots, y_{\mu-1}, y_\mu, y_{\mu+1}, \ldots, y_{\mu+\lambda-1}$ so paroma različni elementi zaporedja $\{y_i\}$. Naj bo $i < j$. Če za elementa y_i in y_j iz zaporedja $\{y_i\}$ velja $y_i = y_j$, potem je $i \geq \mu$. Če označimo $j - i$ z ℓ, potem je zaradi trditve 3.1 število ℓ deljivo z osnovno periodo. Torej je $\ell = k\,\lambda$ za neko naravno število k. Zato je razlika indeksov vsakega para elementov, ki predstavlja trčenje v zaporedju $\{y_i\}$, vedno enaka nekemu večkratniku osnovne periode.

Naj bo G končna neprazna množica. Če izbiramo elemente množice G tako, da je pri vsakem izboru izbran katerikoli element z enako verjetnostjo, potem je to **naključno izbiranje**, izbran element pa imenujemo **naključen element**. Naj bo $f : G \to G$ neka funkcija. Če je $f(g)$ naključen element za vsak $g \in G$, potem je f **naključna funkcija**. Pripomnimo še tole: trditev 3.2 velja tudi v primeru, ko je f naključna funkcija in y_0 naključna začetna vrednost zaporedja $\{y_i\}$. Zato je trditev 3.2 bistvena za razumevanje koncepta delovanja Pollardove ρ-metode.

3.2 Pollardova ρ-metoda

Naj bo G končna neprazna množica. Pollardovo ρ-metodo lahko predstavimo kot metodo, ki temelji na izbiranju elementov iz množice G do trčenja, glej tabelo 3.1. Naj bo $\{y_i\}$ to zaporedje izbranih elementov iz množice G. Elemente

`Vhod:`	končna neprazna množica G.
`Izhod:`	različni števili i in j, za kateri velja $y_i = y_j$.

$i := 0;\ j := 0;$
konec := false;
izberi nek element y iz množice G;
$z[j] := y;$
repeat
  $j := j + 1;$
  izberi nek element y iz množice G;
  if $y = z[s]$ za nek $s \in \{0, \ldots, j-1\}$ **then**
    $i := s;$
    konec := true;
  $z[j] := y;$
until konec
return i, j

Tabela 3.1: Pollardova ρ-metoda.

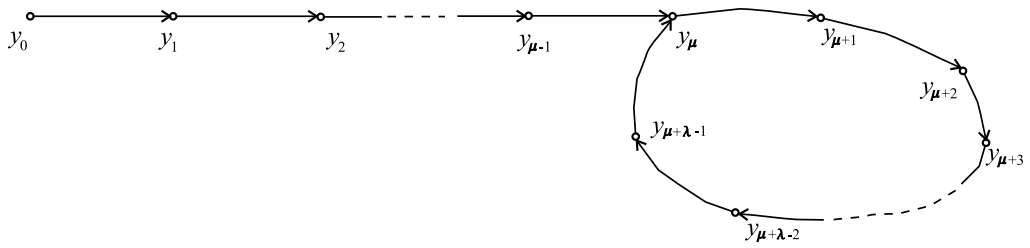
zaporedja $\{y_i\}$ shranjujemo v tabelo. Na j-tem koraku Pollardove ρ-metode izberemo element y_j in preverimo, če ga v tabeli že imamo. Če ga imamo, je prišlo do trčenja in končamo. Sicer y_j dodamo v tabelo. Postopek ponavljamo dokler se ne zgodi trčenje. Do trčenja vedno pride, ker je G končna množica. V najslabšem primeru ponovno izberemo že izbran element po največ $|G|$ korakih. Zato je Pollardova ρ-metoda končen postopek. Naj bosta i in j tisti naravni števili, ki jih vrne Pollardova ρ-metoda, glej tabelo 3.1. Števili i in j določata trčenje v zaporedju $\{y_i\}$, ker je $y_i = y_j$. Število j imenujemo **Pollardov indeks**. Za število i pa velja $i < j$. Vsako zaporedje $\{y_i\}$, ki ga generiramo v Pollardovi ρ-metodi, imenujemo **Pollardovo zaporedje**. Tako definirana Pollardova ρ-metoda v praksi ni uporabna, saj še nismo povedali, kako naj izbiramo elemente iz množice G. To lahko opravimo z računanjem funkcijske vrednosti: $y_{i+1} := f(y_i)$ za $i \geq 0$, kjer je $f : G \to G$ neka funkcija. Le začetni element Pollardovega zaporedja $\{y_i\}$ moramo izbrati na nek način. Tako definirano Pollardovo ρ-metodo podamo v tabeli 3.2. Način iskanja trčenja ostane isti. Po trditvi 3.2

`Vhod:`	končna neprazna množica G, nek začetni element $y_0 \in G$ in neka funkcija $f : G \to G$, ki rekurzivno definira zaporedje $\{y_i\}$.
`Izhod:`	najmanjši različni števili i in j, za kateri velja $y_i = y_j$.

$y := y_0;\ i := 0;\ j := 0;$
konec := false;
$z[j] := y_0;$
repeat
  $j := j + 1;$
  $y := f(y);$
  if $y = z[s]$ za nek $s \in \{0, \ldots, j-1\}$ **then**
    $i := s;$
    konec := true;
  $z[j] := y;$
until konec
return i, j

Tabela 3.2: Pollardova ρ-metoda v praksi.

je zaporedje $\{y_i\}$ periodično. Zato je tudi tako definirana Pollardova ρ-metoda končna. Naj bosta i in j tisti naravni števili, ki jih vrne Pollardova ρ-metoda v tabeli 3.2. Očitno je $i < j \leq |G|$. Naj bo μ osnovna predperioda in λ osnovna perioda zaporedja $\{y_i\}$. Pollardov indeks j je enak $\mu + \lambda$, število i pa je enako μ. Funkcijo f, ki generira zaporedje $\{y_i\}$, imenujemo **iteracijska funkcija**, postopek (3.1) pa **iteracijsko pravilo**. V Pollardovem zaporedju je vsak element (razen začetnega elementa y_0) enak vrednosti iteracijske funkcije na elementu, ki je pred njim. Pollardovemu zaporedju pravimo tudi **zaporedje iteracij** (angl. iterating sequence). Če si posamezne elemente zaporedja $\{y_i\}$ zapisujemo na papir in povežemo zaporedne člene, dobimo obliko grške črke ρ, ki ima na repu μ in na ciklu λ elementov, ki nam pojasni izbiro imena Pollardove ρ-metode, glej sliko 3.1.



Slika 3.1: Zaporedje $\{y_i\}$ v obliki grške črke ρ.

3.3 Prostorska in časovna zahtevnost

Za oceno prostorske in časovne zahtevnosti Pollardove ρ-metode bomo predpostavili, da je Pollardovo zaporedje naključno. To pomeni, da je generirano z naključno iteracijsko funkcijo in z naključno izbranim začetnim elementom. Zato je analiza Pollardove ρ-metode dejansko analiza naključnega zaporedja. Brez te predpostavke ni znano, kako bi le-ta potekala, glej Cohen [5, §8.5.4]. V tem razdelku bomo izračunali število korakov, po katerih je "dovolj" verjetno, da se zgodi vsaj eno trčenje v naključnem zaporedju. Naš cilj je oceniti število korakov, po katerih je verjetnost dogodka vsaj enega trčenja večja ali enaka od nekega vnaprej določenega števila med 0 in 1. Najprej pa bomo navedli nekaj osnovnih definicij, ki jih potrebujemo za računanje verjetnosti naključnega dogodka.

Pogojna verjetnost

Naj bo Ω neprazna množica in $\mathcal{A}$ neka družina dogodkov nad Ω. Naj bodo $A_1, A_2, \ldots, A_k$ poljubni dogodki iz $\mathcal{A}$. Dogodek, da se od dogodkov A_1 in A_2 zgodi vsaj eden, se imenuje **vsota** dogodkov A_1 in A_2. Vsoto dogodkov A_1 in A_2 označimo z $A_1 \cup A_2$. Torej je

$$A_1 \cup A_2 \cup \cdots \cup A_k = \bigcup_{i=1}^{k} A_i$$

dogodek, da se zgodi vsaj eden od dogodkov $A_1, A_2, \ldots, A_k$. Dogodek, da se zgodita dogodka A_1 in A_2 hkrati, se imenuje **produkt** dogodkov A_1 in A_2. Produkt dogodkov A_1 in A_2 označimo z $A_1 \cap A_2$. Če med seboj zmnožimo več dogodkov, dobimo dogodek, da se zgodijo hkrati vsi dogodki $A_1, A_2, \ldots, A_k$ in

pišemo

$$A_1 \cap A_2 \cap \cdots \cap A_k = \bigcap_{i=1}^{k} A_i.$$

Naj bosta dogodka A_1 in A_2 taka, da se oba ne moreta zgoditi v isti ponovitvi poskusa. Taka dogodka imenujemo **nezdružljiva**. Produkt nezdružljivih dogodkov je nemogoč dogodek. Naj bo $A \in \mathcal{A}$ poljuben dogodek. Dogodek, da se ne zgodi A, imenujemo **komplement** dogodka A in ga označimo z $\overline{A}$. Nemogoč dogodek označimo z N, njegov komplement pa z G in ga imenujemo gotov dogodek. Družina dogodkov $\mathcal{A}$ je ***σ*-algebra** nad Ω, če velja:

(i) $\Omega \in \mathcal{A}$;

(ii) če je $A \in \mathcal{A}$, je tudi $\overline{A} \in \mathcal{A}$ in

(iii) če so $A_1, A_2, \ldots$ elementi iz družine $\mathcal{A}$, potem je tudi $\bigcup_{i=1}^{\infty} A_i \in \mathcal{A}$.

Z uporabo lastnosti (i), (ii) in (iii) se lahko hitro prepričamo, da iz $\bigcup_{i=1}^{\infty} A_i \in \mathcal{A}$ sledi, da je tudi $\bigcap_{i=1}^{\infty} A_i \in \mathcal{A}$, ker velja

$$\overline{\left(\bigcup_{i=1}^{\infty} \overline{A_i}\right)} = \bigcap_{i=1}^{\infty} \overline{\overline{A_i}} = \bigcap_{i=1}^{k} A_i.$$

Verjetnostna mera nad družino dogodkov $\mathcal{A}$ je taka funkcija $P : \mathcal{A} \to [0, 1]$, za katero velja $P(G) = 1$ in $P(N) = 0$. Verjetnostna mera priredi vsakemu dogodku $A \in \mathcal{A}$ verjetnost $P(A)$. Za poljubne paroma nezdružljive dogodke $A_1, A_2, \ldots$ iz $\mathcal{A}$ velja

$$P\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} P(A_i).$$

Trojica $(\Omega, \mathcal{A}, P)$ je **verjetnostni prostor**, če je $\mathcal{A}$ σ-algebra nad Ω in P verjetnostna mera nad $\mathcal{A}$. Naj bodo $A_1, A_2, \ldots, A_k$ poljubni dogodki iz $\mathcal{A}$ in $P(A_1) > 0$. Verjetnost, da se zgodi dogodek A_2 pri pogoju, da se je zgodil dogodek A_1, imenujemo **pogojna verjetnost**. Pogojno verjetnost označimo s $P(A_2/A_1)$ in izračunamo z

$$P(A_2/A_1) = \frac{P(A_1 \cap A_2)}{P(A_1)}, \tag{3.2}$$

glej Menezes et al. [30, §2.1.2]. Če odpravimo ulomek v (3.2), dobimo

$$P(A_1 \cap A_2) = P(A_1)\, P(A_2/A_1). \tag{3.3}$$

Naj bo k fiksno naravno število, $A_1, A_2, \ldots, A_k$ poljubni dogodki iz $\mathcal{A}$ in naj bo $P(A_1 \cap A_2 \cap \cdots \cap A_{k-1}) \geq 0$. S popolno indukcijo posplošimo formulo (3.3) na produkt več faktorjev, tj.

$$P(A_1 \cap \cdots \cap A_k) = P(A_1)P(A_2/A_1) \cdots P(A_k/A_1 \cap \cdots \cap A_{k-1}). \tag{3.4}$$

Če v formuli (3.4) nadomestimo dogodke $A_1, \ldots, A_k$ z njihovimi komplementi, dobimo

$$P(\overline{A_k} \cap \cdots \cap \overline{A_1}) = P(\overline{A_1})P(\overline{A_2}/\overline{A_1}) \cdots P(\overline{A_k}/\overline{A_{k-1}} \cap \cdots \cap \overline{A_1}). \tag{3.5}$$

Slednjo formulo lahko uporabimo pri računanju verjetnosti $P(\bigcup_{i=1}^{k} A_i)$, ker velja

$$\overline{\left(\bigcap_{i=1}^{k} \overline{A_i}\right)} = \bigcup_{i=1}^{k} \overline{\overline{A_i}} = \bigcup_{i=1}^{k} A_i. \tag{3.6}$$

Če označimo izraz na levi strani (3.5) z x, potem sledi iz (3.6), da je

$$P\left(\bigcup_{i=1}^{k} A_i\right) = 1 - x,$$

tj. verjetnost, da se zgodi vsaj eden od dogodkov $A_1, \ldots, A_k$, je enaka komplementu verjetnosti, da se zgodijo vsi dogodki $\overline{A_1}, \ldots, \overline{A_k}$. Identiteto (3.6) bomo uporabili v nadaljevanju razdelka.

Verjetnost naključnega dogodka

Sedaj si oglejmo uporabo zgornjih definicij na primeru naključnega izbiranja elementov iz množice G z n elementi. Imamo verjetnostni prostor $(\Omega, \mathcal{A}, P)$, kjer je $\Omega = G$, $\mathcal{A}$ neka družina dogodkov, ki je σ-algebra nad Ω, in P verjetnostna mera. Naključno izbiranje pomeni, da je pri vsaki izbiri z enako verjetnostjo izbran katerikoli element iz množice G. To verjetnost določa velikost množice G. Za vsak $g \in G$ je namreč verjetnost dogodka, da smo naključno izbrali element g iz množice G enaka

$$\frac{1}{n}.$$

Naj bo $A \in \mathcal{A}$ nek dogodek, tj. $A = \{g_1, \ldots, g_k\} \subset G$, kjer je $k \leq n$. Potem lahko izrazimo verjetnost dogodka A z

$$P(A) = \sum_{g \in A} P(\{g\}) = \sum_{g \in A} \frac{1}{n} = \frac{k}{n}.$$

Naj bo k število naključno izbranih elementov iz množice G in $\boldsymbol{P(n,k)}$ verjetnost dogodka, da po k naključnih izbirah elementov iz množice G ni prišlo do trčenja. Verjetnost dogodka vsaj enega trčenja po k korakih izračunamo tako, da od 1 odštejemo $P(n,k)$. Dogodke $A_i \in \mathcal{A}$ definiramo kot dogodke, da se na i-tem koraku naključnega izbiranja zgodi trčenje in zato lahko zapišemo

$$P(n,k) = P(\overline{A_k} \cap \overline{A_{k-1}} \cap \cdots \cap \overline{A_1}).$$

Sedaj nas zanima verjetnost dogodka, da do trčenja po naključnem izbiranju ne pride. Ko izberemo prvi element ($k = 1$), je dogodek, da se trčenje ne zgodi, gotov, tj.

$$P(n,1) = 1.$$

Na drugem koraku ($k = 2$) imamo samo 1 možnost, da je drugi naključno izbran element enak prvemu in zato je verjetnost dogodka, da je drugi naključno izbran element različen od prvega, enaka

$$P(n,2) = 1 - \frac{1}{n}.$$

Ko izbiramo tretji element ($k = 3$), je verjetnost dogodka, da je tretji naključno izbran element enak kateremu od prvih dveh, pri pogoju, da sta prva dva različna, enaka $2/n$. Torej je verjetnost dogodka, da je tretji naključno izbran element različen od prvih dveh, pri pogoju, da sta prva dva različna, enaka $1 - 2/n$. Od prej vemo, da je verjetnost dogodka, da sta prva dva naključno izbrana elementa različna, enaka $1 - 1/n$. Zato lahko zapišemo, da je verjetnost dogodka, da ni trčenja po prvih treh korakih, enaka

$$P(n,3) = \left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right).$$

Ko izberemo četrti element ($k = 4$), je verjetnost dogodka, da je četrti naključno izbran element enak kateremu od prvih treh naključno izbranih elementov, pri pogoju, da so prvi trije elementi različni, enaka $3/n$. Torej je verjetnost dogodka, da je četrti naključno izbran element različen od prvih treh, pri pogoju, da so prvi trije različni, enaka $1 - 3/n$. Sedaj lahko z uporabo zgornje formule zapišemo, da je verjetnost dogodka, da ni trčenja po prvih štirih korakih, enaka

$$P(n,4) = \left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right)\left(1 - \frac{3}{n}\right).$$

Z uporabo oznake za pogojno verjetnost lahko zapišemo, da je pogojna verjetnost dogodkov, da ne pride do trčenja po posameznem koraku, z

$$\begin{aligned} P(\overline{A_1}) &= 1, \\ P(\overline{A_2}/\overline{A_1}) &= 1-\tfrac{1}{n}, \\ P(\overline{A_3}/\overline{A_2}\cap\overline{A_1}) &= 1-\tfrac{2}{n}, \\ P(\overline{A_4}/\overline{A_3}\cap\overline{A_2}\cap\overline{A_1}) &= 1-\tfrac{3}{n}. \end{aligned}$$

Z nadaljevanjem zgornjega razmišljanja in z uporabo formule (3.5) lahko zapišemo, da je verjetnost dogodka, da ni prišlo do trčenja po k korakih, enaka

$$P(n,k) = \left(1-\frac{1}{n}\right)\left(1-\frac{2}{n}\right)\cdots\left(1-\frac{k-1}{n}\right) = \prod_{i=1}^{k-1}\left(1-\frac{i}{n}\right). \tag{3.7}$$

Zaradi formule (3.6) pa lahko zapišemo, da je verjetnost dogodka vsaj enega trčenja po k korakih enaka $1-P(n,k)$. Formulo (3.7) bomo uporabili v naslednjem izreku, katerega dokaz povzamemo po Stinsonu [30, str. 123 in 124]. Najprej bomo z uporabo formule (3.7) izpeljali spodnjo mejo za verjetnost dogodka vsaj enega trčenja po k korakih. S tem bomo dobili diskretno funkcijo spremenljivke k, ki nam za vsak $k \in \mathbb{N}$ pove verjetnost dogodka vsaj enega trčenja po k korakih. Nato bomo to funkcijo za "velike" n navzgor omejili z zvezno funkcijo spremenljivke k. Tako bomo iz te funkcije dobili asimptotično zgornjo mejo za pričakovano število korakov, po katerih se zgodi vsaj eno trčenje s pozitivno verjetnostjo. Z oznako $\approx$ bomo označevali enakost v neskončnosti, z oznako $\doteq$ pa približno enakost.

Izrek 3.3 *Naj bo $\varepsilon \in [0,1)$ ter n in k taki naravni števili, da je*

$$k \geq \sqrt{2\,n\,\ln(1/(1-\varepsilon))}.$$

Potem je za dovolj velike n verjetnost dogodka, da po k naključno izbranih elementih iz množice z n elementi pride do vsaj enega trčenja, vsaj ε, tj.

$$1-P(n,k) \geq \varepsilon.$$

Dokaz. Funkcija $f(x) = 1-x-e^{-x}$ ima v 0 vrednost 0, tj. $f(0)=0$. Za pozitivne x je $f(x)$ padajoča funkcija, tj. $f'(x) = -1+e^{-x} < 0$ za $x > 0$. Od

tod sledi, da je $f(x) \leq 0$ za $x \geq 0$ oziroma $1 - x \leq e^{-x}$. Tako lahko ocenimo $P(n,k)$ iz (3.7) na naslednji način

$$P(n,k) \leq \prod_{i=1}^{k-1} e^{-i/n} = e^{-(1+2+\cdots+(k-1))/n} = e^{-k(k-1)/(2n)}$$

in od tod

$$1 - P(n,k) \geq 1 - e^{-k(k-1)/(2n)}.$$

Za vsako fiksno število $\varepsilon \in [0,1)$ lahko vedno določimo najmanjše tako naravno število k, da je

$$1 - e^{-k(k-1)/(2n)} \geq \varepsilon \quad \text{oziroma} \quad 1 - \varepsilon \geq e^{-k(k-1)/(2n)}.$$

Z logaritmiranjem zgornje neenakosti dobimo

$$\frac{-k(k-1)}{2n} \leq \ln(1-\varepsilon) \quad \text{oziroma} \quad k^2 - k \geq 2n \ln \frac{1}{1-\varepsilon}.$$

Ko gre k v neskončnost je prispevek člena $-k$ zanemarljiv v primerjavi s členom k^2. Zato lahko pišemo $k^2 - k \approx k^2$. Torej pride do vsaj enega trčenja z verjetnostjo najmanj ε, ko k preseže

$$\sqrt{2\,n\,\ln \frac{1}{1-\varepsilon}}.$$

Torej je za vsak fiksen $\varepsilon \in [0,1)$ po najmanj $\sqrt{2\,n\,\ln(1/(1-\varepsilon))}$ korakih verjetnost dogodka vsaj enega trčenja najmanj ε. Z uporabo formule (3.7) dobimo, da je za $k' > k$

$$1 - P(n,k') > 1 - P(n,k).$$

Zato je

$$1 - P(n,k) \geq \varepsilon \quad \text{za} \quad k \geq \sqrt{2\,n\,\ln \frac{1}{1-\varepsilon}}.$$ ∎

Posledica 3.4 *Naj bosta n in k taki naravni števili, da je $k \geq 1,17\sqrt{n}$. Potem je za dovolj velike n verjetnost dogodka, da po k naključno izbranih elementih iz množice z n elementi pride do vsaj enega trčenja, vsaj $1/2$, tj.*

$$1 - P(n,k) \geq \frac{1}{2}.$$

Dokaz. Po predpostavki je $k \geq 1,17\sqrt{n}$. Število $1,17$ je racionalni približek števila $\sqrt{2\ln 2}$. Če vstavimo $\varepsilon = 1/2$ v izrek 3.3, dobimo, da je po $k \geq 1,17\sqrt{n}$ korakih verjetnost dogodka vsaj enega trčenja vsaj $1/2$. ∎

V naslednji trditvi bomo spoznali, zakaj imenujemo izrek 3.3 **paradoks rojstnega dne**.

Trditev 3.5 *Naj bodo elementi množice G vsi dnevi enega leta. Potem je verjetnost dogodka, da obstajata v naključno izbrani množici 23-ih ljudi dva, ki imata rojstni dan na isti dan v letu, več kot $1/2$.*

Dokaz. Če vstavimo $n = 365$ v (3.7), potem dobimo za $k = 23$

$$P(365, 23) = \frac{365}{365}\,\frac{364}{365}\,\frac{363}{365} \cdots \frac{365 - 23 + 1}{365} \doteq 0,493$$

in zato je

$$1 - P(365, 23) \doteq 0,507.$$

Podobno dobimo tudi, če vstavimo $n = 366$ in $k = 23$ v (3.7), tj.

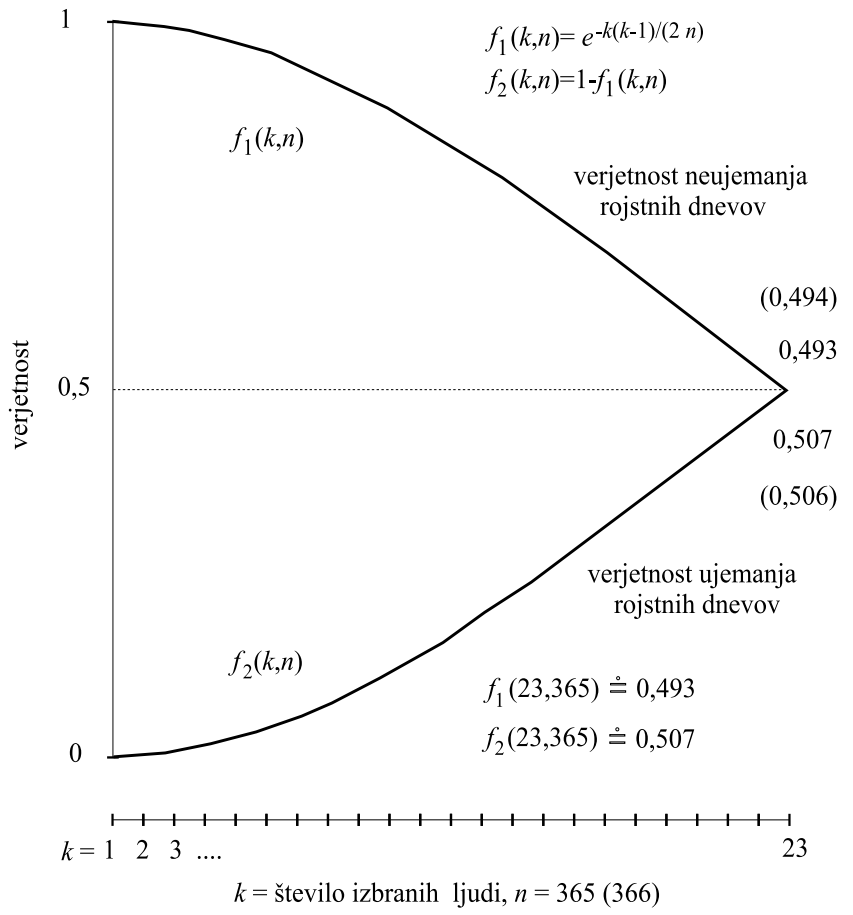
$$P(366, 23) = \frac{366}{366}\,\frac{365}{366}\,\frac{364}{366} \cdots \frac{366 - 23 + 1}{366} \doteq 0,494$$

in zato je

$$1 - P(366, 23) \doteq 0,506.$$ ∎

Trditev 3.5 je lahko veliko presenečenje, ko jo prvič slišimo. To je tudi vzrok, da jo imenujemo paradoks rojstnega dne. Čeprav ima verjetnost dogodka ujemanja in neujemanja rojstnega dne diskretne vrednosti, smo le-to izrazili na sliki 3.2 v obliki grafa funkcije $f_1(k, n) = e^{-k(k-1)/(2\,n)}$, kjer je $k \in [0, 23]$ in $n = 365$ oziroma 366. Zaradi paradoksa rojstnega dne je lažje najti trčenje kot pa trčiti z nekim izbranim elementom. V kriptoanalizi sestavlja paradoks rojstnega dne razred napadov, ki ga imenujemo **napadi s paradoksom rojstnega dne**. Eden izmed njihovih predstavnikov je Pollardova ρ-metoda. O tem nam govori naslednja trditev oziroma njena posledica.

Trditev 3.6 *Naj bo G množica z n elementi in $\{y_i\}$ naključno Pollardovo zaporedje njenih elementov. Potem je $O(\sqrt{n}\,)$ asimptotična zgornja meja za pričakovano število korakov, po katerih se zgodi vsaj eno trčenje v zaporedju $\{y_i\}$ s pozitivno verjetnostjo.*



Slika 3.2: Paradoks rojstnega dne.

Dokaz. Po trditvi 3.2 je zaporedje $\{y_i\}$ periodično, zato je zagotovljena eksistenca nekega trčenja v tem zaporedju. Zanima nas, kakšno je pričakovano število korakov k, po katerih se zgodi vsaj eno trčenje v zaporedju $\{y_i\}$ z neko pozitivno verjetnostjo $\varepsilon \in [0, 1)$. Po predpostavki je zaporedje $\{y_i\}$ naključno. Zato lahko uporabimo izrek 3.3. Naj bo n dovolj veliko število. Potem je $k \geq \sqrt{2\,n\,\ln(1/(1-\varepsilon))} = O(\sqrt{n}\,)$. ∎

Posledica 3.7 *Naj bo G množica z n elementi iz katere naključno izbiramo elemente v Pollardovi ρ-metodi. Potem je $O(\sqrt{n}\,)$ pričakovana asimptotična časovna zahtevnost in tudi pričakovana asimptotična prostorska zahtevnost Pollardove ρ-metode.* ∎

3.4 Pollardova ρ-metoda za DLP

V tem razdelku bomo pokazali, kako je DLP vključen v generiranje Pollardovega zaporedja ter v postopek od trenutka, ko se v njem zgodi trčenje pa do rešitve, glej Pollard [26].

Naj bo g generator grupe G reda n in h nek njen element, katerega diskretni logaritem v osnovi g iščemo. To pomeni, da obstaja natanko določeno število x med 0 in $n-1$, za katerega velja $g^x = h$. Shemo iskanja števila $x = \log_g h$ z uporabo Pollardove ρ-metode podamo v tabeli 3.3. Zaporedje $\{y_i\}$ generiramo

`Vhod:`	množica elementov grupe G in $g, h \in G$.
`Izhod:`	$\log_g h$ ali pa zaključek, da ni rešitve.
	1. Generiranje zaporedja $\{y_i\}$ in iskanje trčenja v njem. 2. V primeru trčenja poskušaj izračunati $\log_g h$.

Tabela 3.3: Pollardova ρ-metoda za DLP.

z nekim začetnim elementom $y_0 \in G$ in z neko iteracijsko funkcijo $f : G \to G$ po pravilu $y_{i+1} = f(y_i)$ za $i \geq 0$. Pri tem moramo izbrati tako iteracijsko funkcijo f, da velja:

če je $y = g^k h^\ell$, kjer sta k in ℓ znani pozitivni celi števili, potem lahko učinkovito izračunamo taki števili k' in ℓ', da velja $f(y) = g^{k'} h^{\ell'}$.

V Pollardovi ρ-metodi za DLP računamo hkrati zaporedje $\{y_i\}$ in taki zaporedji pozitivnih celih števil $\{k_i\}$ in $\{\ell_i\}$, da za vsak $i \geq 0$ velja

$$y_i = g^{k_i} h^{\ell_i}.$$

Zaporedje $\{y_i\}$ in zaporedji pozitivnih celih števil $\{k_i\}$ in $\{\ell_i\}$ zapisujemo v tabelo v obliki

$$(i, y_i, k_i, \ell_i).$$

Na vsakem koraku preverimo, če je element y_i iz tekoče četverice elementov (i, y_i, k_i, ℓ_i) že v tabeli. Recimo, da smo našli trčenje na i-tem koraku. Torej je

$y_i = y_{i'}$ za nek $i' < i$. Potem je

$$g^k\, h^\ell = g^{k'}\, h^{\ell'}.$$

Naj bo $x = \log_g h \bmod n$. Po preureditvi potenc v zgornji enačbi dobimo

$$h^{\ell-\ell'} = g^{k'-k} \tag{3.8}$$

in po logaritmiranju pri osnovi g še linearno kongruenco

$$(\ell - \ell')\, x \equiv (k' - k) \pmod{n}. \tag{3.9}$$

Sedaj je izračun števila x odvisen samo od rešljivosti zgornje linearne kongruence. Oglejmo si, kakšne možnosti imamo

(a) Če $D(\ell - \ell', n) = 1$, potem je

$$\log_g h = (k' - k)(\ell - \ell')^{-1} \bmod n.$$

(b) Če $\ell \equiv \ell' \pmod{n}$, potem linearna kongruenca (3.9) nima rešitve. Nadaljujemo tako, da se vrnemo na prvi korak Pollardove ρ-metode za DLP.

(c) Če $D(\ell - \ell', n) = d > 1$ in $d|(k' - k)$, potem je linearna kongruenca (3.9) rešljiva, sicer se vrnemo na prvi korak Pollardove ρ-metode za DLP.

Sledi natančen opis računanja za primer (c). V kriptosistemih, katerih varnost temelji na težavnosti reševanja DLP oziroma ECDLP, je n večinoma praštevilo ali pa $n = q\,p$, kjer je p veliko praštevilo in q majhno število. To pomeni, da je zelo velika verjetnost, da je d enak 1 in zato primer (c) ni pogost, glej McCurley [21]. Kljub temu si oglejmo računanje v primeru (c). Najprej z uporabo razširjenega Evklidovega algoritma na številih $\ell - \ell'$ in n izračunamo taki števili s in t, da velja

$$d = s(\ell - \ell') + t\, n.$$

Zato je $s(\ell - \ell') \equiv d \pmod{n}$. Če potenciramo enačbo (3.8) na s, dobimo

$$h^d = g^{s(k'-k)}. \tag{3.10}$$

Če želimo nadaljevati z iskanjem števila x, potem mora biti število $s(k'-k)$ oblike $d\,i'$ za neko naravno število i'. Vendar je to možno le, če $d|(k'-k)$. To pa sledi iz (3.9). V slednjem primeru nadaljujemo z računanjem d-tega korena na obeh straneh (3.10). Eksistenca d-tega korena sledi iz tega, ker d deli red grupe. Dobimo

$$h = g^{i'}\, g^{(n/d)j}, \text{ za nek } 0 \leq j \leq d-1.$$

S preverjanjem vseh možnih vrednosti za j (vseh možnih vrednosti za j je d) najdemo takšen $0 \leq j' \leq d-1$, za katerega velja $h = g^{i'}\, g^{(n/d)j'}$. Število j' je enolično določeno, ker je g generator grupe G, h pa nek njen element. Tako lahko zapišemo

$$\log_g h = i' + \frac{n}{d} j'.$$

3.5 Pollardova ρ-metoda za faktorizacijo

Zgodovinsko gledano je bila Pollardova ρ-metoda najprej uporabljena za faktorizacijo sestavljenega števila leta 1975 in šele leta 1978 za računanje diskretnega logaritma, glej Pollard [26]. Do leta 1985 je bil v matematiki prisoten bolj intenziven razvoj teorij na področju problema faktorizacije kot na področju problema diskretnega logaritma, glej §1.3. To je morda tudi eden od glavnih vzrokov, da poznamo danes veliko učinkovitejše metode za faktorizacijo kot je Pollardova ρ-metoda, glej Stinson [30, §5.6.4]. Kljub temu si bomo v tem razdelku ogledali Pollardovo ρ-metodo za faktorizacijo, glej [30, §5.6.2]. Le-ta je namreč učinkovita za izločanje praštevilskih faktorjev velikosti do 10^{30}, tj. do 100-bitnih praštevil. Z najboljšimi algoritmi za faktorizacijo pa lahko v današnjem času faktoriziramo sestavljena števila velikosti do 10^{150}, tj. do 500-bitnih števil.

Naj bo n sestavljeno število in $p \leq \sqrt{n}$ najmanjši (neznan) praštevilski delitelj števila n. To pomeni, da obstajajo netrivialni delitelji števila n. Shematski prikaz iskanja nekega delitelja d števila n z uporabo Pollardove ρ-metode podamo v tabeli 3.4. V Pollardovi ρ-metodi za faktorizacijo postopno generiramo zaporedje $\{y_i\}$ elementov iz množice $\mathbb{Z}_n$. Pri iskanju nekega delitelja d števila n je glavna naloga Pollardove ρ-metode poiskati taki dve števili $x, y \in \{y_i\}$, za kateri velja $x \equiv y \pmod{p}$. To pomeni, da v zaporedju $\{y_i\}$ ne iščemo "eksplicitnega" trčenja, kot smo to počeli v primeru Pollardove ρ-metode za DLP.

`Vhod:`	množica števil $\mathbb{Z}_n$.
`Izhod:`	delitelj d števila n ali pa zaključek, da ni rešitve.
	1. Postopno generiranje zaporedja $\{y_i\}$ in računanje $d = D(y - y', n)$ za vsak par števil y, y' iz $\{y_i\}$. 2. Če $1 < d < n$, vrni d, sicer se vrni na prvi korak.

Tabela 3.4: Pollardova ρ-metoda za faktorizacijo.

Do trčenja po modulu p v zaporedju $\{y_i\}$ pridemo posredno, preko računanja največjega skupnega delitelja med razliko elementov iz $\{y_i\}$ in številom n. Če se to res zgodi, je število d, ki ga vrne Evklidov algoritem na številih $x - y$ in n, ne samo večkratnik praštevila p, ampak tudi netrivialen delitelj števila n. Pollardovo ρ-metodo za faktorizacijo lahko ponovimo na številih d ali n/d, če sta le-ti sestavljeni. Običajno generiramo zaporedje $\{y_i\}$ z neko začetno vrednostjo y_0 in z neko iteracijsko funkcijo $f : \mathbb{Z}_n \to \mathbb{Z}_n$ po pravilu $y_{i+1} = f(y_i)$ za $i \geq 0$. Ker praštevila p ne poznamo, moramo izbrati tako iteracijsko funkcijo f z lastnostjo

$$x \equiv y \pmod{p} \quad \Rightarrow \quad f(x) \equiv f(y) \pmod{p} \quad \text{za} \quad x, y \in \mathbb{Z}_n.$$

V prostoru vseh funkcij iz $\mathbb{Z}_n$ v $\mathbb{Z}_n$ so polinomi primer takih funkcij, ki imajo zgornjo lastnost. Zato je tipično, da izberemo za iteracijsko funkcijo f nek polinom. Izkaže se, da so že kvadratni polinomi dobra izbira. Na primer, $f(x) = x^2 + 1$ in $y_0 = 2$, glej Cohen [5, str. 429]. Kot smo že omenili, računamo elemente zaporedja $\{y_i\}$ vse dokler se ne zgodi $d = D(x - y, n) > 1$ za nek par števil $x, y \in \{y_i\}$. Če je $1 < d < n$, potem smo našli netrivialen delitelj števila n. Če pa je $d = n$, potem moramo poskusiti z drugo izbiro iteracijske funkcije f in začetne vrednosti y_0. Za prikaz delovanja Pollardove ρ-metode za faktorizacijo si oglejmo konkreten primer.

Primer. Število $n = 551$, ki ima delitelja 19 in 29, bomo faktorizirali z uporabo Pollardove ρ-metode. Pollardovo zaporedje $\{y_i\}$ definiramo z iteracijsko funkcijo $f(x) = x^2 + 1 \bmod 551$ in z začetno vrednostjo $y_0 = 2$. Potem za vsak izračunani element $y_k \in \{y_i\}$ izračunamo $d_{j,k} = D(y_k - y_j, 551)$ za vse $0 \leq j < k$ in hkrati preverjamo, če je $d_{j,k} > 1$, glej tabelo 3.5. Iz te tabele vidimo, da

$j \diagdown k$	$y_0 = 2$	$y_1 = 5$	$y_2 = 26$	$y_3 = 126$	$y_4 = 449$
$y_0 = 2$		1	1	1	1
$y_1 = 5$			1	1	1
$y_2 = 26$				1	1
$y_3 = 126$					**19**

Tabela 3.5: Pollardova ρ-metoda za faktorizacijo števila 551 z iteracijsko funkcijo $f(x) = x^2 + 1 \bmod 551$ in z začetno vrednostjo $y_0 = 2$. V tabeli so napisane vrednosti $d_{j,k}$, ki jih vrne Evklidov algoritem na številih $y_k - y_j$ in 551 za $k = 1, \ldots, 4$ in $j = 0, \ldots, k-1$.

smo desetkrat uporabili Evklidov algoritem, da smo našli netrivialen delitelj števila 551. Za $k = 4$ in $j = 3$ smo namreč dobili število $d_{3,4} = 19$, ki je netrivialen delitelj števila 551. Pollardovo zaporedje $\{y_i\}$, ki smo ga izračunali, je $\{2, 5, 26, 126, 449, \ldots\}$ oziroma $\{2, 5, 7, 12, 12, \ldots\}$ po modulu 19. □

Naj bo iteracijska funkcija $f : \mathbb{Z}_n \to \mathbb{Z}_n$ naključna in naj bo p najmanjši praštevilski delitelj števila n. Zanima nas asimptotična zgornja meja za pričakovano število elementov množice $\mathbb{Z}_n$, ki jih moramo naključno izbrati, da najdemo vsaj en tak par $x, y \in \mathbb{Z}_n$ med izbranimi, da je $x \neq y$ in hkrati $x \bmod p = y \bmod p$. Kot smo že povedali, iščemo trčenje v zaporedju števil $\{y_i\}$ po modulu p tako, da uporabimo Evklidov algoritem. Naj bo $\{y_i\}$ to naključno Pollardovo zaporedje elementov iz množice $\mathbb{Z}_n$. Po trditvi 3.6 je $O(\sqrt{p})$ asimptotična zgornja meja za pričakovano število naključno izbranih števil iz množice $\mathbb{Z}_n$ po katerih pride do vsaj enega trčenja med njimi po modulu p s pozitivno verjetnostjo. Po posledici 3.7 pa je $O(\sqrt{p})$ pričakovana asimptotična časovna zahtevnost in tudi pričakovana prostorska zahtevnost Pollardove ρ-metode za faktorizacijo. Ker pa je $p \leq \sqrt{n}$, lahko izrazimo to zahtevnost z $O(n^{1/4})$.

Poglavje 4

Pollard-Floydova ρ-metoda

V Pollardovi ρ-metodi se shranjujejo elementi Pollardovega zaporedja v tabelo. Zato narašča prostorska zahtevnost Pollardove ρ-metode z dolžino Pollardovega zaporedja. Vendar se je temu moč izogniti s primernim algoritmom **iskanja trčenja**. Namen tega poglavja je predstaviti takšne algoritme iskanja trčenja, ki bodo popolnoma odpravili pričakovano eksponentno prostorsko zahtevnost Pollardove ρ-metode, tj. z njihovo uporabo bo postala prostorska zahtevnost Pollardove ρ-metode konstantna. Pri tem pa se pričakovana eksponentna časovna zahtevnost Pollardove ρ-metode ne bo spremenila. V prvem razdelku je opisan Floydov algoritem, v drugem je prikazan Pollardov način reševanja DLP v grupi $(\mathbb{Z}_p^*, *)$, v tretjem pa je predstavljen Brentov algoritem. Od ostalih algoritmov iskanja trčenja je v zadnjem razdelku izpostavljen algoritem **primerjaj in uredi**. Skupni vhodni podatek vsem tem algoritmom je Pollardovo zaporedje $\{y_i\}$ z elementi iz neke končne neprazne množice G. Privzeli bomo, da poznamo začetni element $y_0 \in G$ in iteracijsko funkcijo $f : G \to G$, ki generira to zaporedje $\{y_i\}$ z rekurzijo $y_{i+1} = f(y_i)$ za $i \geq 0$. Kot do sedaj bo μ oznaka za osnovno predperiodo, λ pa oznaka za osnovno periodo zaporedja $\{y_i\}$.

4.1 Floydov algoritem

V tem razdelku si bomo ogledali, kako dosežemo z uporabo Floydovega algoritma konstantno prostorsko zahtevnost Pollardove ρ-metode, glej Knuth [17, naloga 3.1.7].

S Floydovim algoritmom iskanja trčenja v zaporedju $\{y_i\}$ poiščemo najmanjši

takšen indeks j, $1 < j \leq \mu + \lambda$, za katerega velja

$$y_j = y_{2j}. \tag{4.1}$$

Iskanje takšnega indeksa j poteka tako, da sproti računamo zaporedji $\{y_i\}$ in $\{y_{2i}\}$ ter hkrati primerjamo tekoča elementa y_i in y_{2i}, tj. preverjamo veljavnost relacije (4.1). V primeru, če za par elementov (y_i, y_{2i}) ne velja relacija (4.1), izračunamo naslednji par elementov (y_{i+1}, y_{2i+2}) z $y_{i+1} = f(y_i)$ in $y_{2i+2} = f(f(y_{2i}))$ ter spet primerjamo. Postopek ponavljamo, dokler ne najdemo takega para elementov, za katera velja relacija (4.1). Floydov algoritem podamo v tabeli 4.1. Število j, ki ga vrne Floydov algoritem, imenujemo **Flo-**

`Vhod:`	končna neprazna množica G, nek začetni element $y_0 \in G$ in neka iteracijska funkcija $f : G \to G$, ki rekurzivno definira zaporedje $\{y_i\}$.
`Izhod:`	najmanjše število j, za katerega velja $y_j = y_{2j}$.

	$y := y_0;\ z := y_0;\ j := 0;$
	repeat
	  $j := j + 1;$
	  $y := f(y);$
	  $z := f(f(z));$
	until $y = z$
	return j

Tabela 4.1: Floydov algoritem iskanja trčenja v zaporedju $\{y_i\}$.

ydov indeks. Le-ta je odvisen tudi od parametrov μ in λ. Ko se Floydov algoritem konča, ima Floydov indeks eno od spodaj napisanih vrednosti

$$j = \begin{cases} \mu & \text{če} \quad \mu \equiv 0 \pmod{\lambda} \quad \text{in} \quad \mu > 0 \\ \mu + \lambda - (\mu \bmod \lambda) & \text{sicer.} \end{cases} \tag{4.2}$$

Trditev 4.1 *Naj bo G končna množica, $\{y_i\}$ Pollardovo zaporedje njenih elementov in j_0 najmanjše tako naravno število, za katerega velja* (4.1)*. Potem je*

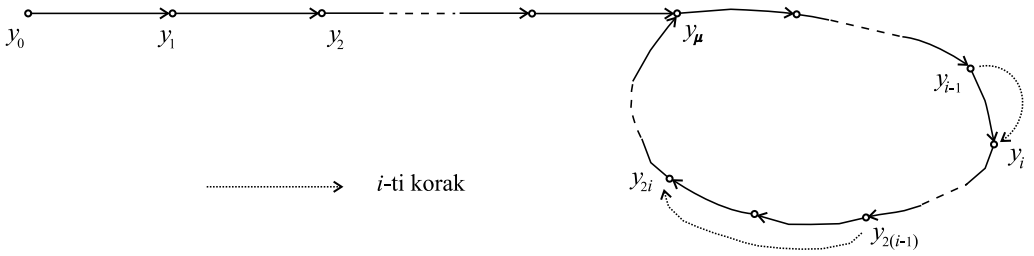
$$j_0 \in \begin{cases} \{\mu, \mu + 1, \ldots, \mu + \lambda - 1\}, & \textit{če} \quad \mu \geq 1 \\ \{\lambda\}, & \textit{sicer.} \end{cases}$$

Dokaz. (Glej sliko 3.1 na strani 51.) S Floydovim algoritmom iščemo najmanjše tako naravno število j_0, za katerega velja (4.1) v zaporedju $\{y_i\}$, tj. iščemo prvo tako trčenje v zaporedju $\{y_i\}$. Očitno je število j_0 nek večkratnik osnovne periode λ zaporedja $\{y_i\}$, glej §3.1. Če je $\mu \geq 1$, potem ima zaporedje $\{y_i\}$ na svojem repu μ elementov. V tem primeru se v najslabšem primeru zgodi prvo trčenje po največ λ korakih, od tedaj, ko pridemo na cikel zaporedja $\{y_i\}$. Zato je $i_0 \in \{\mu, \mu+1, \ldots, \mu+\lambda-1\}$. Če pa je $\mu = 0$, potem je očitno, da se zgodi prvo trčenje po λ korakih. ∎

Naj bo j Floydov indeks. Ko se Floydov algoritem konča, je število izračunanih funkcijskih vrednosti enako $3j$. Označimo to število z W_F. Z upoštevanjem formule (4.2) dobimo za W_F naslednjo oceno

$$3\max(\mu, \lambda) \leq W_F \leq 3(\mu + \lambda). \tag{4.3}$$

Naj bo $(y_i, y_{2i}) \in \{(y_1, y_2), (y_2, y_4), (y_3, y_6), \ldots, (y_i, y_{2i}), \ldots\}$ par elementov, ki ga izračunamo na i-tem koraku Floydovega algoritma. Za izračun para elementov (y_{i+1}, y_{2i+2}) potrebujemo par elementov (y_i, y_{2i}). To pomeni, da potrebujemo s tem načinom iskanja trčenja samo konstanten prostor. En korak Floydovega algoritma podamo na sliki 4.1. V Floydovem algoritmu se razlika indeksov para



Slika 4.1: Floydov algoritem iskanja trčenja v zaporedju $\{y_i\}$.

elementov (y_i, y_{2i}) na vsakem koraku poveča za ena. To pomeni, da se na nekem koraku pojavi taka razlika, ki je enaka nekemu večkratniku osnovne periode λ. Zato je Floydov algoritem končen postopek. Slednji zaključek strnemo v naslednji trditvi, ki je povzeta iz Knuth [17, naloga 3.1.6(b)].

Trditev 4.2 *Naj bo G končna množica in $\{y_i\}$ Pollardovo zaporedje njenih elementov. Potem obstaja najmanjše tako naravno število i, da velja*

$$y_i = y_{2i}.$$

Dokaz. Po trditvi 3.2 je Pollardovo zaporedje periodično, zato je zagotovljena eksistenca trčenja v njem. Naj bo μ osnovna predperioda in λ osnovna perioda zaporedja $\{y_i\}$. Potem je očitno $y_{\mu+\lambda} = y_\mu$ trčenje v zaporedju $\{y_i\}$, glej sliko 3.1 na strani 51. V množici naravnih števil lahko vedno najdemo najmanjše tako število i, za katerega velja

$$i \geq \mu \quad \text{in} \quad i \equiv 0 \pmod{\lambda}.$$

Potem je očitno $y_i = y_{2i}$ trčenje v zaporedju $\{y_i\}$. ∎

Pollardovo ρ-metodo, v kateri uporabimo za iskanje trčenja Floydov algoritem, imenujemo **Pollard-Floydova ρ-metoda**, zaporedje, generirano s to metodo, pa **Pollard-Floydovo** zaporedje. (Vsako Pollard-Floydovo zaporedje je očitno tudi Pollardovo zaporedje.)

Trditev 4.3 *Naj bo G množica z n elementi in $\{y_i\}$ naključno Pollard-Floydovo zaporedje njenih elementov. Potem je $O(\sqrt{n}\,)$ asimptotična zgornja meja za pričakovano število korakov, po katerih se zgodi vsaj eno trčenje v zaporedju $\{y_i\}$ s pozitivno verjetnostjo, pričakovana prostorska zahtevnost pa je konstantna.*

Dokaz. Po trditvi 4.2 je zagotovljena eksistenca trčenja v zaporedju $\{y_i\}$. Oglejmo si, kdaj se to zgodi. Definirajmo $x = \mu + \lambda$, kjer je μ osnovna predperioda in λ osnovna perioda zaporedja $\{y_i\}$. Število korakov Floydovega algoritma, ki jih naredimo do prvega trčenja v zaporedju $\{y_i\}$, je omejeno navzgor s funkcijo $f(x) = x$. Pri tem pa je število izračunanih funkcijskih vrednosti po (4.3) omejeno navzgor s funkcijo $g(x) = 3\,f(x)$. To pomeni, da je Floydov algoritem končen postopek in zaporedje $\{y_i\}$ končno. Sedaj lahko uporabimo še predpostavko, da je zaporedje $\{y_i\}$ naključno. Zato sledi po trditvi 3.6, da je $O(\sqrt{n}\,)$ asimptotična zgornja meja za pričakovano število korakov Floydovega algoritma, po katerih se zgodi vsaj eno trčenje v zaporedju $\{y_i\}$ s pozitivno verjetnostjo. Na vsakem koraku Floydovega algoritma izračunamo nek tekoči par elementov zaporedja $\{y_i\}$. Za to potrebujemo samo nek določen par elementov zaporedja $\{y_i\}$. Zato je pričakovana prostorska zahtevnost konstantna. ∎

Posledica 4.4 *Naj bo G množica z n elementi iz katere naključno izbiramo elemente v Pollard-Floydovi ρ-metodi. Potem je $O(\sqrt{n}\,)$ pričakovana asimptotična časovna zahtevnost Pollard-Floydove ρ-metode, njena pričakovana asimptotična prostorska zahtevnost pa je konstantna.* ∎

4.2 Pollardovo generiranje zaporedja

V tem razdelku si bomo ogledali Pollardov način reševanja DLP v grupi $(\mathbb{Z}_p^*, *)$ [26]. Razdelek bomo zaključili s primerom uporabe le-tega za reševanja DLP v grupi $(\mathbb{Z}_{37}^*, *)$ in v grupi na eliptični krivulji nad obsegom $\mathbb{F}_{101}$.

Naj bo h poljuben element iz grupe $(\mathbb{Z}_p^*, *)$ in g njen generator. Naloga je poiskati diskretni logaritem elementa h v osnovi g v grupi $(\mathbb{Z}_p^*, *)$. Pollard je to nalogo reševal z uporabo Pollard-Floydove ρ-metode. Zaporedje $\{y_i\}$ je definiral s konkretno iteracijsko funkcijo f in s konkretno začetno vrednostjo $y_0 \in \mathbb{Z}_p^*$. Nato je predpostavil, da je f naključna funkcija in y_0 naključen element. Zato je pričakoval najdenje vsaj enega trčenja v zaporedju $\{y_i\}$ po $O(\sqrt{p})$ korakih Pollard-Floydove ρ-metode z neko pozitivno verjetnostjo. Sedaj si oglejmo to definicijo zaporedja $\{y_i\}$. Iteracijsko funkcijo $f_P : \mathbb{Z}_p^* \to \mathbb{Z}_p^*$ je Pollard definiral s

$$f_P(y) = \begin{cases} h\,y & ; \quad 0 < y \leq p/3, \\ y^2 & ; \quad p/3 < y \leq 2p/3, \\ g\,y & ; \quad 2p/3 < y < p. \end{cases} \tag{4.4}$$

Funkcijo f_P imenujemo **Pollardova iteracijska funkcija**. Za začetno vrednost zaporedja $\{y_i\}$ je izbral $y_0 = 1$. Hkrati z zaporedjem $\{y_i\}$ je definiral taki zaporedji pozitivnih celih števil $\{\alpha_i\}$ in $\{\beta_i\}$, za kateri velja

$$y_i = h^{\alpha_i}\, g^{\beta_i} \quad \text{za} \quad i = 0, 1, 2, \ldots .$$

Iz $y_0 = 1$ sledi, da sta začetni vrednosti zaporedij $\{\alpha_i\}$ in $\{\beta_i\}$ enaki $\alpha_0 = 0$ in $\beta_0 = 0$. Zveza med zaporednimi členi zaporedij $\{\alpha_i\}$ in $\{\beta_i\}$ pa je

$$\alpha_{j+1} = \begin{cases} \alpha_i + 1 & ; \quad 0 < y_i \leq p/3, \\ 2\alpha_i & ; \quad p/3 < y_i \leq 2p/3, \\ \alpha_i & ; \quad 2p/3 < y_i < p \end{cases}$$

in

$$\beta_{j+1} = \begin{cases} \beta_i & ; \quad 0 < y_i \leq p/3, \\ 2\beta_i & ; \quad p/3 < y_i \leq 2p/3, \\ \beta_i + 1 & ; \quad 2p/3 < y_i < p, \end{cases}$$

kjer elemente zaporedij $\{\alpha_i\}$ in $\{\beta_i\}$ računamo po modulu $(p-1)$. Število $p-1$ je namreč red osnove g diskretnega logaritma v grupi $(\mathbb{Z}_p^*, *)$ in tudi velikost te grupe. Z reševanjem DLP je nadaljeval tako, da je za iskanje trčenja v

zaporedju $\{y_i\}$ uporabil Floydov algoritem. Na vsakem njegovem koraku je izračunal šesterico elementov

$$(y_i, \alpha_i, \beta_i, y_{2i}, \alpha_{2i}, \beta_{2i})$$

in hkrati preverjal enakost elementov y_i in y_{2i}. Naj bo $x = \log_g h \bmod (p-1)$. V primeru trčenja je poskušal izračunati x z reševanjem linearne kongruence

$$(\beta_i - \beta_{2i})\, x \equiv (\alpha_{2i} - \alpha_i) \pmod{(p-1)}. \tag{4.5}$$

Kako rešujemo linearno kongruenco (4.5), smo že opisali v §3.4. Za ilustracijo si oglejmo konkreten primer uporabe Pollardovega načina reševanja DLP.

Primer. Izračunali bomo diskretni logaritem števila 17 v osnovi 5 v grupi $(\mathbb{Z}_{37}^*, *)$. V tem primeru je $h = 17$ in $g = 5$. Naj bo $S_1 = \{1, 2, \ldots, 12\}$, $S_2 = \{13, 14, \ldots, 24\}$ in $S_3 = \{25, 26, \ldots, 36\}$ delitev množice števil $\mathbb{Z}_{37}^*$. Pollardovo iteracijsko funkcijo f_P definiramo s

$$f_P(y) = \begin{cases} 17\,y & ; \quad y \in S_1, \\ y^2 & ; \quad y \in S_2, \\ 5\,y & ; \quad y \in S_3. \end{cases}$$

Nato definiramo taki zaporedji pozitivnih celih števil $\{\alpha_i\}$ in $\{\beta_i\}$, da velja $y_i = 17^{\alpha_i}\, 5^{\beta_i}$ za $i \geq 0$, z

$$\alpha_{i+1} = \begin{cases} \alpha_i + 1 & ; \quad y_i \in S_1, \\ 2\alpha_i & ; \quad y_i \in S_2, \\ \alpha_i & ; \quad y_i \in S_3 \end{cases} \quad \text{in} \quad \beta_{i+1} = \begin{cases} \beta_i & ; \quad y_i \in S_1, \\ 2\beta_i & ; \quad y_i \in S_2, \\ \beta_i + 1 & ; \quad y_i \in S_3, \end{cases}$$

kjer elemente zaporedij števil $\{\alpha_i\}$ in $\{\beta_i\}$ računamo po modulu 36. Ker je $y_0 = 1$, sta začetni vrednosti zaporedij števil $\{\alpha_i\}$ in $\{\beta_i\}$ enaki 0, tj. $\alpha_0 = 0$ in $\beta_0 = 0$. V tabeli 4.2 podamo šesterice elementov $(y_i, \alpha_i, \beta_i, y_{2i}, \alpha_{2i}, \beta_{2i})$, kot jih dobimo do dogodka prvega trčenja. Kot vidimo iz tabele 4.2, je prišlo do trčenja na šestem koraku. Tako smo ugotovili, da je $3 \equiv 17^6\, 5^4 \pmod{37}$ in tudi $3 \equiv 17^{32}\, 5^{18} \pmod{37}$. Naj bo $x = \log_5 17 \bmod 36$. Če izenačimo oba zapisa števila 3 ter to potem še preuredimo, dobimo $17^{26} \equiv 5^{22} \pmod{37}$. Če slednje logaritmiramo pri osnovi 5, dobimo linearno kongruenco $26\,x \equiv 22 \pmod{36}$, glej (4.5). Rešitev le-te je $x = 5 \bmod 36$. Torej je $\mathbf{\log_5 17 = 5}$ v grupi $(\mathbb{Z}_{37}^*, *)$. □

i	y_i	α_i	β_i	y_{2i}	α_{2i}	β_{2i}
0	1	0	0	1	0	0
1	17	1	0	30	2	0
2	30	2	0	34	3	1
3	2	2	1	3	6	4
4	34	3	1	11	14	8
5	22	3	2	34	16	8
6	**3**	6	4	**3**	32	18

Tabela 4.2: Računanje diskretnega logaritma števila 17 v osnovi 5 v grupi $(\mathbb{Z}_{37}^*, *)$ z uporabo Pollard-Floydove ρ-metode.

Na enak način kot je Pollard reševal DLP v grupi $(\mathbb{Z}_p^*, *)$, ga lahko rešujemo tudi v grupi na eliptični krivulji nad končnim obsegom. Oglejmo si slednje na konkretnem primeru.

Primer. Naj bo eliptična krivulja definirana nad obsegom $\mathbb{F}_{101}$ z enačbo $y^2 = x^3+17\,x+1$. Naj bo $G = (E_{17,1}(\mathbb{F}_{101}), +)$ grupa na dani eliptični krivulji, P neka njena točka in k neko celo število. Npr. z uporabo programskega paketa Pari, ki je dosegljiv na internetnem naslovu (`http://www.parigp-home.de/`), izračunamo število elementov v grupi G. Dobimo $|G| = 103$. Zato je $k\,P = (k \bmod 103)P$ za vsak $P \in G$ in za vsako celo število k. Vzemimo točko $P = (0, 1) \in G$ za osnovo diskretnega logaritma v grupi G. Ker je grupa G praštevilskega reda, je vsak njen element, ki je različen od $\mathcal{O}$, hkrati tudi generator grupe G. Izberimo še eno njeno točko, npr. $Q = (5, 98)$ in zastavimo nalogo.

Poišči tako celo število k med 0 in 102, da bo veljalo $Q = k\,P$, tj. $k = \log_P Q \bmod 103$. Diskretni logaritem k računamo po modulo 103. Naj bo $x(R)$ oznaka za x koordinato poljubne točke R iz grupe G. Število k bomo izračunali z uporabo Pollardovega načina generiranja zaporedja $\{Y_i\}$ v grupi G po Pollard-Floydovi ρ-metodi. Naj bo $S_1 = \{0, 1, \ldots, 33\}$, $S_2 = \{34, 35, \ldots, 67\}$ in $S_3 = \{68, 69, \ldots, 100\}$ delitev množice števil $\mathbb{F}_{101}$. Zaporedje $\{Y_i\}$ generiramo z začetnim elementom $Y_0 = \mathcal{O}$ in s Pollardovo iteracijsko funkcijo f_P, ki jo definiramo s

$$f_P(Y) = \begin{cases} (5, 98) + Y & ; \quad x(Y) \in S_1, \\ 2\,Y & ; \quad x(Y) \in S_2, \\ (0, 1) + Y & ; \quad x(Y) \in S_3. \end{cases}$$

Hkrati z zaporedjem $\{Y_i\}$ definiramo tudi taki zaporedji pozitivnih celih števil

$\{\alpha_i\}$ in $\{\beta_i\}$, da velja $Y_i = \alpha_i\, Q + \beta_i\, P$ za $i \geq 0$, z

$$\alpha_{i+1} = \begin{cases} \alpha_i + 1 & ; \ x(Y_i) \in S_1, \\ 2\alpha_i & ; \ x(Y_i) \in S_2, \\ \alpha_i & ; \ x(Y_i) \in S_3 \end{cases} \quad \text{in} \quad \beta_{i+1} = \begin{cases} \beta_i & ; \ x(Y_i) \in S_1, \\ 2\beta_i & ; \ x(Y_i) \in S_2, \\ \beta_i + 1 & ; \ x(Y_i) \in S_3, \end{cases}$$

kjer elemente zaporedij števil $\{\alpha_i\}$ in $\{\beta_i\}$ računamo po modulu 103 (število 103 je namreč red grupe G in zato tudi red vsakega njenega generatorja, v tem primeru točke $P = (0, 1)$). Ker je $Y_0 = \mathcal{O}$, sta začetni vrednosti zaporedij števil $\{\alpha_i\}$ in $\{\beta_i\}$ enaki 0, tj. $\alpha_0 = 0$ in $\beta_0 = 0$. V tabeli 4.3 podamo šesterice elementov $(Y_i, \alpha_i, \beta_i, Y_{2i}, \alpha_{2i}, \beta_{2i})$ do prvega trčenja. Po kratkem računanju ugo-

i	Y_i	α_i	β_i	Y_{2i}	α_{2i}	β_{2i}
0	$\mathcal{O}$	0	0	$\mathcal{O}$	0	0
1	(5,98)	1	0	(68,60)	2	0
2	(68,60)	2	0	(12,32)	4	2
3	(63,29)	2	1	(97,77)	6	2
4	(12,32)	4	2	(53,81)	12	6
5	(8,89)	5	2	(62,66)	24	13
6	**(97,77)**	6	2	**(97,77)**	96	52

Tabela 4.3: Računanje diskretnega logaritma elementa $(5, 98)$ v osnovi $(0, 1)$ v grupi $(E_{17,1}(\mathbb{F}_{101}), +)$ z uporabo Pollard-Floydove ρ-metode.

tovimo, da je $(97, 77) = 6\, Q + 2\, P$ in tudi $(97, 77) = 96\, Q + 52\, P$. Naj bo $x = \log_{(0,1)}(5, 98) \bmod 103$. Če izenačimo oba zapisa točke $(97, 77)$ ter to potem še preuredimo, dobimo $-90\, Q = 50\, P$. Ker je $-90 \equiv 13 \pmod{103}$, sledi, da je $13\, Q = 50\, P$. Če slednje logaritmiramo pri osnovi P, dobimo linearno kongruenco $13\, x \equiv 50 \pmod{103}$, glej (4.5). Rešitev le-te je $x = 91 \bmod 103$. Torej je $\mathbf{\log_P Q = 91}$ v grupi $(E_{17,1}(\mathbb{F}_{101}), +)$. □

4.3 Brentov algoritem

V Floydovem algoritmu izračunamo na vsakem koraku tri funkcijske vrednosti, glej tabelo 4.1. Vendar lahko dosežemo z uporabo Brentovega algoritma računanje samo ene funkcijske vrednosti, glej Brent [2] in Cohen [5, str. 427-429].

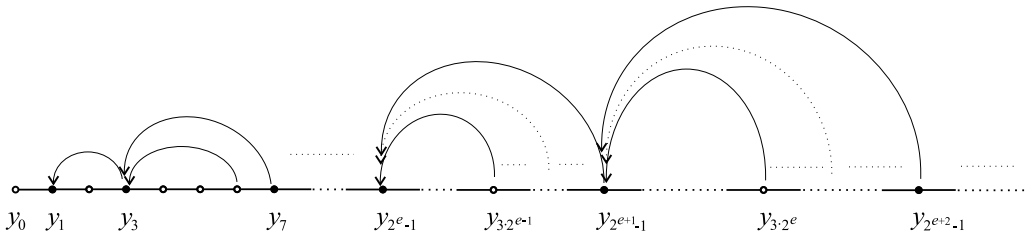
Naj bo G končna neprazna množica, $\{y_i\}$ Pollardovo zaporedje njenih elementov in $e \geq 1$. V Brentovem algoritmu računamo zaporedje $\{y_i\}$ z rekurzijo $y_{i+1} = f(y_i)$ za $i \geq 0$, kjer je $f : G \to G$ neka iteracijska funkcija in $y_0 \in G$ nek začetni element. Trčenje v zaporedju $\{y_i\}$ iščemo takole: vsakič, ko pridemo do elementa z indeksom $2^e - 1$, si ga zapomnimo. Nato primerjamo element y_{2^e-1} z elementi

$$y_{2^e+k-1} \quad \text{za} \quad 2^{e-1} < k \leq 2^e.$$

Če se trčenje ne zgodi, tj. $y_{2^e-1} \neq y_{2^e+k-1}$ za vse $k \in \{2^{e-1}+1, 2^{e-1}+2, \ldots, 2^e\}$, je $y_{2^{e+1}-1}$ zadnji izračunani element. Sedaj elementa y_{2^e-1} ne potrebujemo več. Zato ga lahko "pozabimo". Na njegovo mesto v spominu postavimo element $y_{2^{e+1}-1}$, ki ga uporabimo za primerjanje z elementi

$$y_{2^{e+1}+k-1} \quad \text{za} \quad 2^e < k \leq 2^{e+1}.$$

Postopek ponavljamo, dokler se ne zgodi trčenje. Pripomnimo, da v Brentovem algoritmu začnemo iskati trčenje tako, da si najprej zapomnimo element $y_1 := f(y_0)$, ki ga nato primerjamo z elementom y_3. Grafičen prikaz iskanja trčenja v Pollardovem zaporedju $\{y_i\}$ z Brentovim algoritmom podamo na sliki 4.2. Na tej sliki označujejo črne pike tiste elemente iz zaporedja $\{y_i\}$, ki jih imamo



Slika 4.2: Iskanje trčenja v zaporedju $\{y_i\}$ z Brentovim algoritmom.

v trenutnem spominu na določenih korakih Brentovega algoritma. Bele pike označujejo vse ostale elemente, ki jih računamo, loki s puščico pa povezujejo tiste elemente, ki jih primerjamo. Kot vidimo na sliki 4.2, nam določenih primerjav v Brentovem algoritmu ni potrebno narediti. Razlog, zakaj lahko določene primerjave v Brentovem algoritmu izpustimo, si oglejmo v naslednjem odstavku.

Naj bo $e \geq 1$. Privzemimo, da primerjamo elemente $y_{2^e}, y_{2^e+1}, \ldots, y_{2^{e+1}-1}$ zaporedja $\{y_i\}$ z elementom $y_{2^e-1} \in \{y_i\}$ in da se trčenje po $2^e - 1$ korakih

ne zgodi. Brez izgube na splošnosti lahko privzamemo, da je zaporedje $\{y_i\}$ brez repa, tj. $\mu = 0$. Potem lahko zaključimo, da osnovna perioda λ zaporedja $\{y_i\}$ ni e-bitno število. $2^e - 1$ je namreč največje e-bitno število. Ker ni prišlo do trčenja, je $y_{2^{e+1}-1}$ naslednji element, ki si ga zapomnimo in primerjamo z elementi $y_{2^{e+1}}, \ldots, y_{2^{e+2}-1}$. Sedaj pa nam primerjave v $2^e - 1$ korakih ni potrebno narediti. Razlika med indeksi elementa $y_{2^{e+1}-1}$ in elementov $y_{2^{e+1}}, \ldots, y_{3\cdot 2^e-2}$ je namreč manjša ali enaka $2^e - 1$. Slednjih primerjav nam torej ni potrebno narediti zato, ker za osnovno periodo λ vemo, da je vsaj $(e+1)$-bitno število. 2^e je namreč najmanjše $(e+1)$-bitno število. Zato nadaljujemo s primerjavami vključno od elementa $y_{2^{e+1}+2^e-1} = y_{3\cdot 2^e-1}$ naprej. To vsekakor upoštevamo v Brentovem algoritmu, katerega shematski prikaz podamo v tabeli 4.4. Število

`Vhod:`	končna neprazna množica G, nek začetni element $y_0 \in G$ in neka iteracijska funkcija $f : G \to G$, ki rekurzivno definira zaporedje $\{y_i\}$.
`Izhod:`	različni števili i in j, za kateri velja $y_i = y_j$.

```
y := y_0; r := 1; j := 0;
konec := false;
repeat
  w := y;
  i := j;                    % i = 2^{e-1} - 1 za nek e ∈ ℕ
  r := 2r;
  repeat
    j := j + 1;              % j je tekoči indeks
    y := f(y);
    if j ≥ 3/4 r then
      if w = y then
        konec := true
  until j ≥ r - 1 or konec
until konec
return i, j
```

Tabela 4.4: Brentov algoritem iskanja trčenja v zaporedju $\{y_i\}$.

j, ki ga vrne Brentov algoritem, imenujemo **Brentov indeks**. Število i, ki ga

tudi vrne Brentov algoritem, pa je neka potenca števila dve minus ena. Leto označuje indeks tistega elementa, ki ga imamo v spominu, ko se Brentov algoritem konča. Naj bo $e \geq 2$ število ponovitev zunanje (*repeat-until*) zanke Brentovega algoritma, ko se le-ta konča. Potem je $i = 2^{e-1} - 1$, glej tabelo 4.4. Naj bo λ osnovna perioda zaporedja $\{y_i\}$ in ℓ najmanjše tako naravno število, da je $\ell\,\lambda \geq 2^{e-2} + 1$. Brentov algoritem iskanja trčenja lahko predstavimo kot algoritem, v katerem iščemo takšen Brentov indeks j,

$$j = 2^{e-1} + \ell\,\lambda - 1 \leq 2^e - 1, \tag{4.6}$$

za katerega velja

$$y_{2^{e-1}-1} = y_j.$$

Ko se Brentov algoritem konča, nam pove Brentov indeks j število izračunanih funkcijskih vrednosti, število narejenih primerjav pa je navzgor omejeno z

$$2^{e-1} - 1. \tag{4.7}$$

Naj bo μ osnovna predperioda zaporedja $\{y_i\}$ in $\overline{e}$ najmanjše tako naravno število, za katerega velja

$$2^{\overline{e}-1} - 1 \geq \max(\mu, \lambda).$$

Ker je $2^{\overline{e}} - 2^{\overline{e}-1} - 1 = 2^{\overline{e}-1} - 1 \geq \lambda$, sledi, da je $e \leq \overline{e}$. Zato je $i \leq 2\max(\mu, \lambda)$. Torej lahko ocenimo Brentov indeks j z

$$j = i + \ell\,\lambda \leq 2\max(\mu, \lambda) + \ell\,\lambda \leq 2\mu + (\ell + 2)\,\lambda. \tag{4.8}$$

Razlika indeksov para elementov $(y_{2^e-1},\, y_{2^e+k-1})$, kjer je $k \in \{2^{e-1}+1, 2^{e-1}+2, \ldots, 2^e\}$, se na vsakem koraku poveča za ena. Ta razlika narašča od $2^{e-1}+1$ do 2^e. Če se trčenje ne zgodi, se na naslednjem sklopu korakov razlika indeksov para elementov $(y_{2^{e+1}-1}, y_{2^{e+1}+k-1})$, kjer je $k \in \{2^e+1, 2^e+1, \ldots, 2^{e+1}\}$, povečuje prav tako po ena. Vendar pa narašča od $2^e + 1$ do 2^{e+1}. To pomeni, da se na nekem koraku pojavi taka razlika, ki je enaka nekemu večkratniku osnovne periode λ. Zato je Brentov algoritem končen postopek.

Trditev 4.5 *Naj bo G končna množica in $\{y_i\}$ Pollardovo zaporedje njenih elementov. Potem obstaja najmanjše tako naravno število e in najmanjše tako število $k \geq 2^{e-1} + 1$, da velja*

$$y_{2^e-1} = y_{2^e+k-1}.$$

Dokaz. Po trditvi 3.2 je Pollardovo zaporedje periodično, zato je zagotovljena eksistenca trčenja . Naj bosta i in j taki naravni števili, $i < j$, da je $y_i = y_j$ trčenje v zaporedju $\{y_i\}$. Velja $j - i = v\,\lambda$, kjer je v neko naravno število in λ osnovna perioda zaporedja $\{y_i\}$. Naj bo e' najmanjše tako število, da velja $2^{e'} - 1 \geq i$. V zaporedju naravnih števil $2^{e'-1} + 1, 2^{e'-1} + 2, \ldots, 2^{e'-1} + j - i - 1$ vedno obstaja najmanjše tako število k', ki je deljivo z λ (vseh števil v zaporedju $2^{e'-1} + 1, 2^{e'-1} + 2, \ldots, 2^{e'-1} + j - i - 1$ je najmanj λ, v splošnem pa jih je lahko nek večkratnik števila λ). Potem je $y_{2^{e'}-1} = y_{2^{e'}+k'-1}$ trčenje v zaporedju $\{y_i\}$ za taki števili e' in k'. Naj bo μ osnovna predperioda zaporedja $\{y_i\}$. Sedaj pa lahko izberemo tudi najmanjše tako število e, da velja $2^{e-1} \leq \max(\mu, \lambda) < 2^e$. Ker je $\lambda \leq 2^e - 1$, obstaja na intervalu $[2^e + 1, 2^{e+1}]$ najmanjše tako število k, ki je deljivo z λ. Torej je

$$y_{2^e-1} = y_{2^e+k-1}$$

za taki števili e in k. ∎

Pollard-Brentovo ρ-metodo, v kateri uporabimo za iskanje trčenja Brentov algoritem, imenujemo **Pollard-Brentova ρ-metoda**, zaporedje, generirano s to metodo, pa **Pollard-Brentovo** zaporedje.

Trditev 4.6 *Naj bo G množica z n elementi in $\{y_i\}$ naključno Pollard-Brentovo zaporedje njenih elementov. Potem je $O(\sqrt{n}\,)$ asimptotična zgornja meja za pričakovano število korakov, po katerih se zgodi vsaj eno trčenje v zaporedju $\{y_i\}$ s pozitivno verjetnostjo, pričakovana prostorska zahtevnost pa je konstantna.*

Dokaz. Po trditvi 4.5 je zagotovljena eksistenca trčenja v zaporedju $\{y_i\}$. Oglejmo si, kdaj se to zgodi. Naj bo $e \geq 2$ število ponovitev zunanje (*repeat-until*) zanke Brentovega algoritma, ko se le-ta konča, μ najmanjša predperioda, λ najmanjša perioda zaporedja $\{y_i\}$ in $\ell \geq 1$ najmanjše tako število, da je $\ell\,\lambda \geq 2^{e-2} + 1$. Definirajmo $x = 2\mu + (\ell + 2)\,\lambda$. Število korakov Brentovega algoritma, ki jih naredimo do prvega trčenja v zaporedju $\{y_i\}$, je po (4.6) omejeno navzgor z $2^e - 1$. Število narejenih primerjav v zaporedju $\{y_i\}$ je po (4.7) omejeno navzgor z $2^{e-1} - 1$, število izračunanih funkcijskih vrednosti pa je po (4.8) omejeno navzgor s funkcijo $f(x) = x$. To pomeni, da je Brentov algoritem končen postopek. Sedaj lahko uporabimo še predpostavko, da je zaporedje $\{y_i\}$ naključno. Zato sledi po trditvi 3.6, da je $O(\sqrt{n}\,)$ asimptotična zgornja meja za

pričakovano število korakov Brentovega algoritma, po katerih se zgodi vsaj eno trčenje v zaporedju $\{y_i\}$ s pozitivno verjetnostjo. Na vsakem koraku Brentovega algoritma izračunamo en element zaporedja $\{y_i\}$. Za to potrebujemo samo element, ki je pred njim v zaporedju $\{y_i\}$. Hkrati pa imamo vedno samo en dodaten element zaporedja $\{y_i\}$ v spominu, ki ga uporabljamo za primerjanje. Zato je pričakovana prostorska zahtevnost konstantna. ∎

Posledica 4.7 *Naj bo G množica z n elementi iz katere naključno izbiramo elemente v Pollard-Brentovi ρ-metodi. Potem je $O(\sqrt{n}\,)$ pričakovana asimptotična časovna zahtevnost Pollard-Brentove ρ-metode, njena pričakovana asimptotična prostorska zahtevnost pa je konstantna.* ∎

4.4 Algoritem primerjaj in uredi

V do sedaj omenjenih algoritmih iskanja trčenja v tem poglavju smo potrebovali spomin za največ dva elementa Pollardovega zaporedja. Zato je pričakovana prostroska zahtevnost teh algoritmov konstantna. Ta pričakovana prostorska zahtevnost pa bi bila še vedno konstantna, če bi npr. dopustili uporabiti neko fiksno velikost spomina. V tem razdelku si bomo ogledali algoritem primerjaj in uredi (angl. Compare and Adjust), ki uporablja "večji" spomin, a ima še vedno konstantno pričakovano prostorsko zahtevnost, glej Teske [31, str. 41-48].

Ideja

Naj bo G končna neprazna množica, $\{y_i\}$ Pollardovo zaporedje njenih elementov, μ osnovna predperioda in λ osnovna perioda zaporedja $\{y_i\}$. Potem je $y_{\sigma+\lambda} = y_\sigma$ trčenje v zaporedju $\{y_i\}$ za vsako naravno število $\sigma \geq \mu$. Naj bo v najmanjše tako naravno število, da velja $\mu + \lambda \leq v\,\mu$. Potem je $\sigma + \lambda \leq v\,\sigma$ za vsako naravno število $\sigma \geq \mu$. Torej, če si zapomnimo element $y_\sigma \in \{y_i\}$ in ga primerjamo z elementi $y_{\sigma+1}, y_{\sigma+2}, \ldots$, potem zagotovo najdemo trčenje $y_{\sigma+\lambda} = y_\sigma$ v zaporedju $\{y_i\}$. Pri tem velja $\mu \leq \sigma < \sigma + \lambda \leq v\,\sigma$.

Algoritem primerjaj in uredi

V algoritmu primerjaj in uredi rekurzivno računamo elemente Pollardovega zaporedja $\{y_i\}$. Naj bosta v in t taki naravni števili, za kateri velja $v, t \geq 2$ in $(v-1)|t$. Število t določa število elementov v spominu, število v pa določa

`Vhod:`	končna neprazna množica G, nek začetni element $y_0 \in G$ in neka iteracijska funkcija $f : G \to G$, ki rekurzivno definira zaporedje $\{y_i\}$.
`Izhod:`	različni števili i in j, za kateri velja $y_i = y_j$.

$y := y_0;\ j := 0;\ i := 0;$
konec := false;
$\sigma_1 := 0; \ldots; \sigma_t := 0;$
$z[\sigma_1] := y_0; \ldots; z[\sigma_t] := y_0;$
repeat
  $j := j + 1;$
  $y := f(y);$
  if $y = z[\sigma_s]$ za nek σ_s, kjer je $s \in \{1, \ldots, t\}$ **then** % primerjaj
    $i := \sigma_s;$
    konec := true;
  if $j \geq v\,\sigma_1$ **then** % uredi
    $z[\sigma_1] := z[\sigma_2]; \ldots; z[\sigma_{t-1}] := z[\sigma_t];$
    $z[\sigma_t] := y;$
    $\sigma_1 := \sigma_2; \ldots; \sigma_{t-1} := \sigma_t;$
    $\sigma_t := j;$
until konec
return i, j

Tabela 4.5: Iskanja trčenja v zaporedju $\{y_i\}$ z algoritmom primerjaj in uredi.

tiste elemente zaporedja $\{y_i\}$, ki jih shranimo na določenem koraku algoritma primerjaj in uredi v spomin. Naj bodo $\sigma_1, \ldots, \sigma_t$ oznake indeksov elementov, ki jih imamo v spominu, tj. v neki tabeli Z, ki ima prostor za t elementov iz množice G. Na začetku algoritma napolnimo tabelo Z z začetnim elementom y_0, tj. $Z[i] := y_0$ za $1 \leq i \leq t$. Zato so vsi indeksi elementov v tabeli na začetku enaki nič, tj. $\sigma_1 = \cdots = \sigma_t = 0$. Potem rekurzivno računamo elemente zaporedja $\{y_i\}$ in hkrati preverjamo, če je izračunani element $y_j \in \{y_i\}$ v tabeli Z, tj. preverjamo ali je $y_j = Z[s]$ za kak $s \in \{1, \ldots t\}$. Če je, smo našli trčenje in končamo. Sicer preverimo, če je njegov indeks večji ali enak $v\,\sigma_1$. Če je, ciklično premaknemo elemente tabele Z za eno mesto v levo in na njeno skrajno

desno mesto postavimo tekoči element, tj. $Z[t] := y_j$. Postopek ponavljamo do trčenja. Shematski prikaz algoritma primerjaj in uredi podamo v tabeli 4.5. Števili v in t imenujemo **parametra** algoritma primerjaj in uredi. Število j, ki ga vrne algoritem primerjaj in uredi, imenujemo **indeks trčenja**. Število i, ki ga tudi vrne algoritem, pa označuje indeks tistega elementa iz tabele Z, s katerim smo trčili. Parametra v in t določata trčenje $y_\sigma = y_{\sigma+\lambda}$ v zaporedju $\{y_i\}$, kjer je

$$\sigma \leq \left(1 + \frac{v-1}{t}\right) \max\left(\frac{\lambda}{v-1}, \mu\right),$$

glej Teske [31, izrek 3.4]. Iz tega lahko ocenimo zgornjo mejo za vrednost indeksa trčenja j z

$$j \leq \max(2\mu, \lambda) + \lambda \leq 2(\mu + \lambda),$$

glej [31, (3.14)]. Pollard-Floydovo ρ-metodo, v kateri uporabimo za iskanje trčenja algoritem primerjaj in uredi s parametroma v in t, imenujemo **(v, t)-Pollardova ρ-metoda**, zaporedje, generirano s to metodo, pa **(v, t)-zaporedje**. Podobno kot pri dokazu trditve 4.5 lahko tudi tu pokažemo, da obstaja v vsakem (v, t)-zaporedju $\{y_i\}$ trčenje, tj. vsako (v, t)-zaporedje $\{y_i\}$ je končno. Naj bo G množica z n elementi in $\{y_i\}$ naključno (v, t)-zaporedje njenih elementov. Potem lahko pokažemo zelo podobno kot pri dokazu trditve 4.6, da je $O(\sqrt{n}\,)$ asimptotična zgornja meja za pričakovano število korakov, po katerih se zgodi vsaj eno trčenje v zaporedju $\{y_i\}$ s pozitivno verjetnostjo. Pri tem je pričakovana prostorska zahtevnost konstantna. Zato je $O(\sqrt{n}\,)$ pričakovana asimptotična časovna zahtevnost (v, t)-Pollardove ρ-metode, njena pričakovana asimptotična prostorska zahtevnost pa je konstantna.

Trditev 4.8 *Naj bo G množica z n elementi in $\{y_i\}$ naključno (v, t)-zaporedje njenih elementov. Potem je $O(\sqrt{n}\,)$ asimptotična zgornja meja za pričakovano število korakov, po katerih se zgodi vsaj eno trčenje v zaporedju $\{y_i\}$ s pozitivno verjetnostjo, pričakovana prostorska zahtevnost pa je konstantna.* ∎

Posledica 4.9 *Naj bo G množica z n elementi iz katere naključno izbiramo elemente v (v, t)-Pollardovi ρ-metodi. Potem je $O(\sqrt{n}\,)$ pričakovana asimptotična časovna zahtevnost (v, t)-Pollardove ρ-metode, njena pričakovana asimptotična prostorska zahtevnost pa je konstantna.* ∎

Z uporabo teorije verjetnosti lahko predstavimo vsako lastnost naključnega zaporedja z neko funkcijo, ki jo imenujemo **slučajna spremenljivka**. Vsaki taki funkciji lahko določimo **pričakovano vrednost** oziroma **matematično upanje** (angl. expectation), ki ga označujemo z E, glej Menezes et al. [23, §2.1.3]. V tabeli 4.6 podamo pričakovano vrednost indeksa trčenja j, pričakovano število primerjav in pričakovano prostorsko zahtevnost za vse algoritme iskanja trčenja v Pollardovem zaporedju, ki smo jih predstavili v tem delu.

ρ-metoda	$E(j)$	E(primerjav)	E(prostor)	reference
Pollardova	$1,2533\sqrt{n}$	$O(n)$	$1,2533\sqrt{n}$	[17, naloga 3.1.12]
Pollard-Floydova	$1,0308\sqrt{n}$	$1,0308\sqrt{n}$	$O(1)$	[26]
Pollard-Brentova	$2,2393\sqrt{n}$	$0,8832\sqrt{n}$	$O(1)$	[5, trditev 8.5.6]
$(3,8)$-Pollardova	$1,4591\sqrt{n}$	$11,5932\sqrt{n}$	$O(1)$	[31, (3.15)]

Tabela 4.6: Pričakovana vrednost indeksa trčenja j, pričakovano število primerjav in pričakovana prostorska zahtevnost za vse algoritme iskanja trčenja v Pollardovem zaporedju, ki smo jih predstavili v tem delu. V zadnjem stolpcu so reference na literaturo, kjer so te pričakovane vrednosti izračunane ali pa samo navedene. Število n označuje velikost množice, v kateri rekurzivno računamo elemente Pollardovega zaporedja do nekega trčenja.

Poglavje 5

Pollardova zaporedja

V Pollardovi ρ-metodi nimamo natančno predpisanega postopka, po katerem generiramo naključno Pollardovo zaporedje, glej §3.2. Zato je namen tega poglavja predstavitev konkretnih načinov za generiranje naključnega Pollardovega zaporedja. Vsako tako zaporedje generiramo v neki končni in neprazni množici G. Če se omejimo na Pollardovo ρ-metodo za reševanje DLP (glej §3.4), nam je lahko pri generiranju naključnega zaporedja v veliko pomoč Pollardov način, ki je predstavljen v §4.2. Njegov način generiranja zaporedja v Pollardovi ρ-metodi lahko posplošimo na

(i) naključno izbiro začetnega elementa $y_0 \in G$ zaporedja $\{y_i\}$,

(ii) delitev množice G na $t \geq 3$ "približno" enako velikih delov $S_1, \ldots, S_t$, tj. $S_1 \cup \cdots \cup S_t = G$ in $S_i \cap S_j = \emptyset$ za $i \neq j$ in

(iii) izbiro iteracijske funkcije $f : G \to G$, ki jo lahko definiramo kot $f = (f_1, \ldots, f_t)$, kjer $f_i : S_i \to G$ za $i = 1, \ldots, t$.

Pri vsaki računalniški implementaciji Pollardove ρ-metode za reševanje DLP moramo izbrati začetni element zaporedja $\{y_i\}$. Zato točke (i) ne bomo posebej obravnavali. Bolj se bomo posvetili ostalima točkama. Točki (ii) bomo namenili prvi razdelek tega poglavja, točki (iii) pa drugega. Osredotočili se bomo na delitve množice števil končnega obsega in množice točk eliptične krivulje nad končnim obsegom. Nato bomo definirali iteracijske funkcije na definiranih delitvah.

5.1 Delitev množice

Ogledali si bomo delitve množice števil praštevilskega obsega $\mathbb{F}_p$, $p \geq 3$. Te delitve lahko posplošimo na delitve množice števil obsega $\mathbb{F}_{p^n}$, kjer je n naravno število. Nato si bomo ogledali delitve množice števil obsega $\mathbb{F}_{2^n}$. V zadnjem delu razdelka bomo definirali delitve množice točk na eliptični krivulji nad končnim obsegom.

Delitve množice števil obsega $\mathbb{F}_p$, $p \geq 3$

Naj bo G množica števil praštevilskega obsega $\mathbb{F}_p$, tj. $G = \{0, 1, 2, \ldots, p-1\}$. Prva delitev množice G je

$$\begin{aligned} S_1 &= \{x \in G \,|\, 0 \leq x \leq \tfrac{p}{3}\}, \\ S_2 &= \{x \in G \,|\, \tfrac{p}{3} < x < \tfrac{2p}{3}\}, \\ S_3 &= \{x \in G \,|\, \tfrac{2p}{3} \leq x < p\}. \end{aligned} \tag{5.1}$$

Množice S_1, S_2 in S_3 delitve (5.1) so paroma disjunktne in enako velike. Če upoštevamo ostanke števil iz množice G po deljenju s 3, dobimo delitev

$$\begin{aligned} S_1 &= \{x \in G \,|\, x \equiv 1 \pmod 3\}, \\ S_2 &= \{x \in G \,|\, x \equiv 2 \pmod 3\}, \\ S_3 &= \{x \in G \,|\, x \equiv 0 \pmod 3\}. \end{aligned} \tag{5.2}$$

Ker ima vsako naravno število natanko en ostanek po deljenju s 3, so množice S_1, S_2 in S_3 delitve (5.2) paroma disjunktne. Ker ima vsako zaporedje treh zaporednih naravnih števil tri različne ostanke po deljenju s 3, so množice S_1, S_2 in S_3 delitve (5.2) enako velike. Kot smo že omenili lahko delitvi (5.1) in (5.2) posplošimo na delitev množice števil $\mathbb{F}_{p^n}$, kjer je n naravno število. V nadaljevanju si bomo ogledali delitve množice števil obsega sode karakteristike.

Delitve množice števil obsega $\mathbb{F}_{2^n}$

Naj bo G množica števil obsega $\mathbb{F}_{2^n}$, kjer je n naravno število. Množico G lahko delimo na oba načina kot smo to naredili za množico števil praštevilskega

obsega $\mathbb{F}_p$, $p \geq 3$. Delitev (5.1) se v tem primeru glasi takole

$$\begin{aligned} S_1 &= \{x \in G \,|\, 0 \leq x \leq \tfrac{2^n}{3}\}, \\ S_2 &= \{x \in G \,|\, \tfrac{2^n}{3} < x < \tfrac{2^{n+1}}{3}\}, \\ S_3 &= \{x \in G \,|\, \tfrac{2^{n+1}}{3} \leq x < 2^n\}. \end{aligned} \tag{5.3}$$

Množice S_1, S_2 in S_3 delitve (5.3) so paroma disjunktne in enako velike. Delitev (5.2) pa lahko predstavimo v tem primeru tako, da upoštevamo kriterij deljenja s 3 za števila v binarnem zapisu. Vsak element $a \in G$ lahko predstavimo v obliki vsote potenc števila 2 oziroma z nekim zaporedjem ničel in enk dolžine n, tj.

$$a = \sum_{i=0}^{n-1} a_i \, 2^i = (a_{n-1}a_{n-2} \ldots a_1 a_0)_2,$$

kjer $a_i \in \{0, 1\}$ za $0 \leq i \leq n-1$. Potence 2^j, kjer je $j \in \mathbb{N}_0$, imajo pri deljenju s 3 naslednjo lastnost

$$2^j \equiv \begin{cases} 1 \pmod 3, & \text{če je j sodo število,} \\ -1 \pmod 3, & \text{če je j liho število.} \end{cases}$$

Pokažimo to lastnost. Če je $j = 2k$, kjer je k neko naravno število, velja

$$2^j - 1 = (2^2 - 1)(2^{2(k-1)} + 2^{2(k-2)} + \cdots + 1) \equiv 0 \pmod 3$$

in zato je $2^j \equiv 1 \pmod 3$. Če je pa $j = 2k + 1$, velja

$$2^j + 1 = (2 + 1)(2^{j-1} - 2^{j-2} + \cdots + 1) \equiv 0 \pmod 3$$

in zato je $2^j \equiv -1 \pmod 3$. Zaradi zgornje lastnosti, in ker je vsota kongruenc enaka kongruenci vsote, lahko določimo ostanek po deljenju s 3 vsakemu elementu $a \in G$ z uporabo naslednje formule

$$a \bmod 3 = \left(\sum_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} a_{2i} - \sum_{i=0}^{\lfloor \frac{n-1}{2} \rfloor} a_{2i+1} \right) \bmod 3. \tag{5.4}$$

Torej se delitev (5.2) v tem primeru z upoštevanjem formule (5.4) glasi takole

$$\begin{aligned} S_1 &= \{a \in G \,|\, a \bmod 3 = 1\}, \\ S_2 &= \{a \in G \,|\, a \bmod 3 = 2\}, \\ S_3 &= \{a \in G \,|\, a \bmod 3 = 0\}. \end{aligned} \tag{5.5}$$

Množice S_1, S_2 in S_3 delitve (5.5) so paroma disjunktne in enako velike. Delitev (5.5) je zelo primerna za računalniško implementacijo. Za določitev ostanka po deljenju s 3 potrebujemo namreč z uporabo formule (5.4) samo nekaj taktov računalniške ure. Z uporabo binarnega zapisa lahko predstavimo še dve delitvi množice G, ki sta prav tako primerni za računalniško implementacijo. Definirajmo funkcijo $wt : G \to \mathbb{N} \cup \{0\}$ z $wt(a) =$ število enk, ki jih ima število a v binarnem zapisu. Funkcijo wt imenujemo **Hammingova utežna funkcija** (angl. Hamming weight function), $wt(a)$ pa **Hammingova utež** števila $a \in G$, glej Menezes et al. [23, opomba 3.59]. Prva delitev množice G z uporabo funkcije wt je

$$\begin{aligned} S_1 &= \{a \in G \,|\, wt(a) \equiv 1 \pmod 3\}, \\ S_2 &= \{a \in G \,|\, wt(a) \equiv 2 \pmod 3\}, \\ S_3 &= \{a \in G \,|\, wt(a) \equiv 0 \pmod 3\}. \end{aligned} \tag{5.6}$$

Ker velja enoličen zapis števil v obliki vsote potenc števila 2, so množice S_1, S_2 in S_3 delitve (5.6) paroma disjunktne in enako velike. Druga delitev množice G z uporabo funkcije wt je

$$\begin{aligned} S_1 &= \{a \in G \,|\, 0 \le wt(a) \le \tfrac{n}{3}\}, \\ S_2 &= \{a \in G \,|\, \tfrac{n}{3} < wt(a) < \tfrac{2n}{3}\}, \\ S_3 &= \{a \in G \,|\, \tfrac{2n}{3} \le wt(a) \le n\}. \end{aligned} \tag{5.7}$$

Množice S_1, S_2 in S_3 delitve (5.7) so paroma disjunktne, niso pa enako velike. Da je to res, se lahko hitro prepričamo na primeru. Naj bo G množica števil obsega $\mathbb{F}_{2^9}$. Če delimo množico G po pravilu (5.7), dobimo množice naslednjih velikosti: $|S_1| = 130$, $|S_2| = 252$ in $|S_3| = 130$. Torej je očitno, da množice S_1, S_2 in S_3 delitve (5.7) niso enako velike.

Delitev množice točk na eliptični krivulji nad obsegom $\mathbb{F}_p$, $p > 3$

Naj bo E množica točk na eliptični krivulji nad praštevilskim obsegom $\mathbb{F}_p$, $p > 3$. Elementi množice E so vsi pari števil $(X, Y) \in \mathbb{F}_p \times \mathbb{F}_p$, ki ustrezajo neki dani enačbi eliptične krivulje, skupaj s posebno točko $\mathcal{O}$, glej §1.3. Naj bo $\{Y_i\}$ Pollardovo zaporedje z elementi iz množice E. Pri vsaki delitvi množice E moramo paziti na to, kateri množici iz delitve pripada točka $\mathcal{O}$. S tem lahko

namreč preprečimo naslednje stanje v generiranju Pollardovega zaporedja $\{Y_i\}$: če je $Y_i = \mathcal{O}$ za nek $i > 0$ in imamo iteracijsko funkcijo $f : E \to E$, definirano s $f(Y) = 2\,Y$, potem je $Y_j = \mathcal{O}$ za vsak $j > i$, ker je $2\,\mathcal{O} = \mathcal{O}$. Kateri množici iz delitve pripada točka $\mathcal{O}$ določimo šele potem, ko izberemo iteracijsko funkcijo. Naj bo $E = E - \{\mathcal{O}\}$ množica točk eliptične krivulje in $P = (X, Y) \in E$ neka njena točka. Definirajmo naslednje funkcije $x : E \to \mathbb{F}_p$ z $x(P) = X$, $y : E \to \mathbb{F}_p$

$P = (X,Y) \in E$	$P \in S_1$	$P \in S_2$	$P \in S_3$
$x(P) = X$	$0 \le X \le \frac{p}{3}$	$\frac{p}{3} < X < \frac{2p}{3}$	$\frac{2p}{3} \le X < p$
$y(P) = Y$	$0 \le Y \le \frac{p}{3}$	$\frac{p}{3} < Y < \frac{2p}{3}$	$\frac{2p}{3} \le Y < p$
$u(P) = X + Y$	$0 \le X + Y \le \frac{p}{3}$	$\frac{p}{3} < X + Y < \frac{2p}{3}$	$\frac{2p}{3} \le X + Y < p$

Tabela 5.1: Delitve množice točk na eliptični krivulji nad obsegom $\mathbb{F}_p$, $p > 3$, z uporabo delitve (5.1), ki je Pollardova delitev množice števil obsega $\mathbb{F}_p$.

z $y(P) = Y$ in $u : E \to \mathbb{F}_p$ z $u(P) = X + Y$. Velja $u(P) = x(P) + y(P)$. Z upoštevanjem delitve (5.1) in z uporabo funkcij $x(\cdot), y(\cdot)$ in $u(\cdot)$ dobimo tri delitve množice E, ki jih podamo v tabeli 5.1. V vseh treh delitvah so množice S_1, S_2 in S_3 paroma disjunktne, ker so definirane z uporabo funkcij $x(\cdot), y(\cdot)$ in $u(\cdot)$. Ali so množice S_1, S_2 in S_3 v vseh treh delitvah enako velike, pa je odprto vprašanje. Za vsako eliptično krivuljo se lahko s štetjem točk v množicah S_1, S_2 in S_3 prepričamo, ali so le-te enako velike. Množico E lahko delimo tudi z upoštevanjem delitve (5.2). Tako dobimo nove tri delitve množice E, ki jih podamo v tabeli 5.2. V vseh treh delitvah so množice S_1, S_2 in S_3 paroma

$P = (X,Y) \in E$	$P \in S_1$	$P \in S_2$	$P \in S_3$
$x(P) = X$	$X \equiv 1$	$X \equiv 2$	$X \equiv 0$
$y(P) = Y$	$Y \equiv 1$	$Y \equiv 2$	$Y \equiv 0$
$u(P) = X + Y$	$X + Y \equiv 1$	$X + Y \equiv 2$	$X + Y \equiv 0$

Tabela 5.2: Delitve množice točk na eliptični krivulji nad obsegom $\mathbb{F}_p$, $p > 3$, z uporabo delitve (5.2), ki je delitev množice števil obsega $\mathbb{F}_p$ glede na ostanek po deljenju s 3.

disjunktne, ker ima vsako število natanko en ostanek po deljenju s 3. Ali so množice S_1, S_2 in S_3 v vseh treh delitvah enako velike, je tudi odprto vprašanje.

Podobno kot pri delitvah iz tabele 5.1, se lahko tudi tu za vsako eliptično krivuljo s štetjem točk v množicah S_1, S_2 in S_3 prepričamo, ali so le-te enako velike.

Delitev množice točk na eliptični krivulji nad $\mathbb{F}_p$ na več kot tri dele

Naj bo E množica točk na eliptični krivulji nad praštevilskim obsegom $\mathbb{F}_p$, $p > 3$, A neko racionalno število med 0 in 1 in $b : E \to \mathbb{R}$ neka injektivna funkcija. Oglejmo si delitev množice E na $r \geq 3$ delov, glej Teske [31, str. 38 in 39]. Definirajmo funkcijo $f^* : E \to [0, 1)$ s

$$f^*(P) = \begin{cases} (A \cdot b(P)) \bmod 1 & \text{če } P \neq \mathcal{O} \\ 0 & \text{če } P = \mathcal{O}, \end{cases}$$

kjer je $(A \cdot b(P)) \bmod 1 = A \cdot b(P) - \lfloor A \cdot b(P) \rfloor$. Definirajmo še funkcijo $f : E \to \{1, \ldots, r\}$ s

$$f(P) = \lfloor f^*(P)\, r \rfloor + 1.$$

Funkcija f nam razdeli množico E na r delov. To delitev definiramo takole

$$S_t = \{P \in E : f(P) = t\}, \text{ za } 1 \leq t \leq r. \tag{5.8}$$

Množice S_t delitve (5.8) so po konstrukciji paroma disjunktne. Domneva se, da so množice S_t, $1 \leq t \leq r$, delitve (5.8) približno enako velike tudi zaradi konstrukcije.

Naj bo število $\tau \in \mathbb{R}$ pozitivna realna rešitev enačbe $x^2 - x - 1 = 0$. Število τ imenujemo število zlatega razmerja[1] (angl. golden ratio) in je

$$\tau = \frac{\sqrt{5} + 1}{2} = 1,6180339\ldots.$$

Velja $\tau^{-1} = \tau - 1 = 0,6180339\ldots$. Pogosto imenujemo tudi število τ^{-1} število zlatega razmerja, glej Knuth [18, str. 510]. Če vzamemo v definiciji funkcije f^* za racionalno število A nek racionalen približek števila τ^{-1}, potem dobimo "najbolj" enakomerno porazdelitev funkcijskih vrednosti $f^*(P)$, kjer je $P \in E$, na intervalu $[0, 1)$, glej [18, slika 37 in izrek S]. Zato lahko domnevamo, da dobimo s takim postopkom "najbolj" enakomerno porazdelitev točk množice E v množice S_t, $1 \leq t \leq r$. Če domneva drži, potem lahko res trdimo, da so

[1] Naj bosta a in b poljubni realni števili in naj velja $a < b$. Če razdelimo interval $[a, b]$ na dva dela tako, da je razmerje dolžin večjega in manjšega dela enako razmerju $b - a$ in večjega dela, pravimo, da smo ga razdelili v zlatem razu, glej Razpet [27].

množice S_t, $1 \leq t \leq r$, približno enako velike. Ko enkrat določimo injektivno funkcijo $b : E \to \mathbb{R}$ in željeno število množic $r \geq 3$ v delitvi množice E, je vrednost funkcije $f(P)$, kjer je $P \in E$, odvisna samo od števila decimalnih mest števila τ^{-1}, ki ga izberemo za racionalno število A v definiciji funkcije f^*. Definirajmo najmanjše število decimalnih mest števila τ^{-1} pri danih b in r kot tisto število decimalnih mest, od katerega vsako naslednje ne vpliva na vrednost funkcije $f(P) \in \{1, \ldots, r\}$ za vsako točko $P \in E$. Sedaj bomo določili to najmanjše število decimalnih mest števila τ^{-1}.

Naj bo c neko pozitivno realno število in $\|b\|$ norma preslikave b, tj. $\|b\| := \max\{b(P) : P \in E\}$. Potem je očitno, da vpliva samo prvih $\lceil \log_{10} r \rceil$ decimalnih mest števila $c \bmod 1$ na vrednost $\lfloor (c \bmod 1)\, r \rfloor$. To pomeni, da samo prvih $\lceil \log_{10} \|b\| \rceil + \lceil \log_{10} r \rceil$ decimalnih mest števila τ^{-1}, ki ga izberemo za racionalno število A, še določa vrednost funkcije $f(P)$ za vsako točko $P \in E$, tj. še vpliva na vrednost $\lfloor ((A \cdot b(P)) \bmod 1)\, r \rfloor$. Torej je $2 + \lfloor \log_{10}(\|b\|\, r) \rfloor$ najmanjše število decimalnih mest števila τ^{-1}, ker je

$$\lceil \log_{10} \|b\| \rceil + \lceil \log_{10} r \rceil \leq 2 + \lfloor \log_{10}(\|b\|\, r) \rfloor.$$

Slednje velja zaradi lastnosti logaritma v realnih številih. Za vsaki realni števili x in y velja namreč

$$\lceil \log_{10} x \rceil \leq \lfloor \log_{10} x \rfloor + 1 \quad \text{in} \quad \lfloor \log_{10} x \rfloor + \lfloor \log_{10} y \rfloor \leq \lfloor \log_{10}(x\, y) \rfloor.$$

Na zelo podoben način, kot smo delili množice točk na eliptični krivulji nad praštevilskim obsegom $\mathbb{F}_p$, $p > 3$, lahko delimo tudi množice točk eliptične krivulje nad obsegom $\mathbb{F}_{p^n}$ in $\mathbb{F}_{2^n}$, kjer je n naravno število. Zato teh delitev ne bomo posebej obravnavali.

5.2 Iteracijske funkcije

V Pollardovi ρ-metodi za DLP je priporočljivo izbrati tako iteracijsko funkcijo, ki bo generirala Pollardovo zaporedje $\{y_i\}$ naključno. Potem lahko namreč pričakujemo najdenje vsaj enega trčenja v zaporedju $\{y_i\}$ po $O(\sqrt{n}\,)$ korakih Pollardove ρ-metode, kjer je n velikost množice v kateri rešujemo DLP, glej §3.3. Prvo tako primerno iteracijsko funkcijo smo spoznali v §4.2, kjer smo predstavili

Pollardov način generiranja zaporedja $\{y_i\}$ v grupi $(\mathbb{Z}_p^*, *)$, kjer je p praštevilo. V §3.4 pa smo spoznali, da hkrati z računanjem zaporedja $\{y_i\}$ računamo tudi taki zaporedji pozitivnih celih števil $\{\alpha_i\}$ in $\{\beta_i\}$, da velja $y_i = h^{\alpha_i} g^{\beta_i}$, kjer je g osnova diskretnega logaritma in h element, katerega diskretni logaritem v osnovi g iščemo. V tem razdelku bomo hkrati z definicijo vsake iteracijske funkcije predstavili tudi pravila za generiranje zaporedij števil $\{\alpha_i\}$ in $\{\beta_i\}$. Predstavili bomo iteracijske funkcije, ki jih lahko uporabimo v multiplikativni grupi, nato jih bomo predstavili še za aditivno grupo. Tipičen predstavnik slednjih grup, ki jih v kriptografiji pogosto uporabljamo, so grupe na eliptični krivulji nad končnim obsegom.

Posplošitev Pollardove iteracijske funkcije f_P

Naj bosta g in h elementa končne neprazne množice G in S_1, S_2 in S_3 neka delitev množice G. Iščemo diskretni logaritem elementa h v osnovi g. Za generiranje zaporedja $\{y_i\}$ lahko uporabimo Pollardovo iteracijsko funkcijo (4.4) oziroma njeno posplošitev definirano s

$$f_P(y) = \begin{cases} h\,y & \text{če } y \in S_1, \\ y^2 & \text{če } y \in S_2, \\ g\,y & \text{če } y \in S_3. \end{cases} \tag{5.9}$$

Zaporedji pozitivnih celih števil $\{\alpha_i\}$ in $\{\beta_i\}$ računamo po modulu $|G|$ in po pravilu:

$$\alpha_{j+1} = \begin{cases} \alpha_i + 1 & y_i \in S_1 \\ 2\alpha_i & y_i \in S_2 \\ \alpha_i & y_i \in S_3 \end{cases} \quad \text{in} \quad \beta_{j+1} = \begin{cases} \beta_i & y_i \in S_1 \\ 2\beta_i & y_i \in S_2 \\ \beta_i + 1 & y_i \in S_3. \end{cases} \tag{5.10}$$

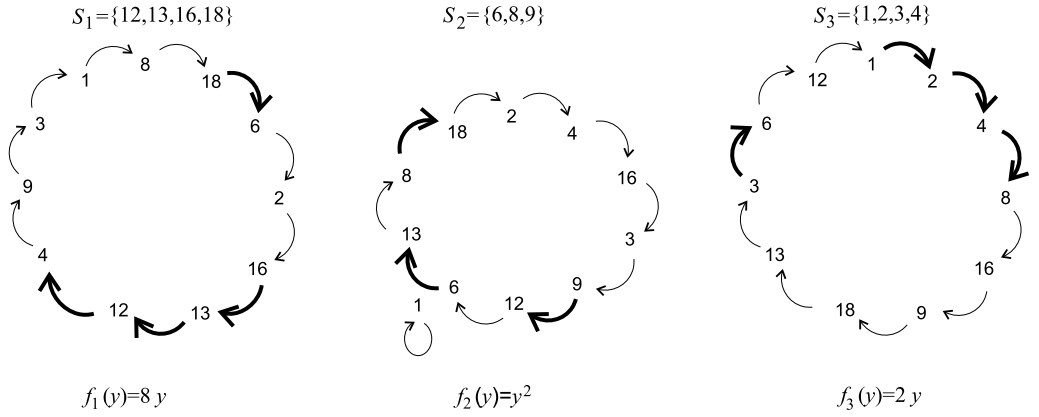
Pollardovo iteracijsko funkcijo (4.4) oziroma njeno posplošitev (5.9) lahko zapišemo kot $f_P = (f_1, f_2, f_3)$, kjer $f_i : S_i \to G$ za $i = 1, 2$ in 3. V primeru, da je G

$f_P = (f_1, f_2, f_3)$	$G = \mathbb{F}_p$	$G \subseteq \mathbb{F}_p$, $\lvert G\rvert$ praštevilo
f_1	vedno	vedno
f_2	nikoli	vedno
f_3	vedno	vedno

Tabela 5.3: Bijektivnost Pollardove iteracijske funkcije f_P.

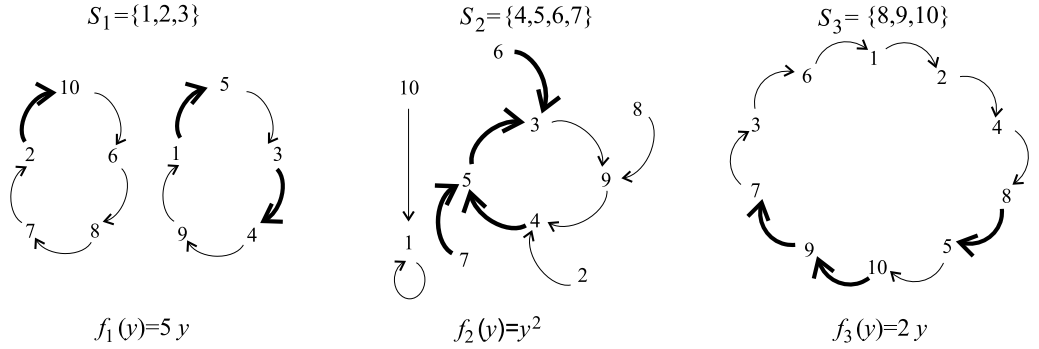
množica števil obsega $\mathbb{F}_p$ in S_1, S_2 in S_3 neka delitev množice G, potem lahko

hitro povemo, kdaj je katera od preslikav f_i bijektivna. Slednje podamo v tabeli 5.3. Kot vidimo iz tabele 5.3, sta funkciji f_1 in f_3 vedno bijektivni. Funkcija f_2 pa je bijektivna samo v primeru, če je množica G praštevilske moči. Grafičen prikaz Pollardove iteracijske funkcije (5.9) na množici števil grupe $\langle 2\rangle \subseteq (\mathbb{Z}_{23}^*, *)$ podamo na sliki 5.1. Na tej sliki se lahko hitro prepričamo, da so funkcije f_i res bijektivne. Za primerjavo si oglejmo Pollardovo iteracijsko funkcijo (4.4) na



Slika 5.1: Posplošena Pollardova iteracijska funkcija na množici števil grupe $\langle 2\rangle \subseteq (\mathbb{Z}_{23}^*, *)$ s $h = 8$ in $g = 2$.

množici števil grupe $(\mathbb{Z}_{11}^*, *)$, ki je podana na sliki 5.2. V tem primeru funkcija f_2 ni bijektivna. Zato je uporaba take funkcije primernejša za generiranje naključnega zaporedja v Pollardovi ρ-metodi.



Slika 5.2: Pollardova iteracijska funkcija $f_P = (f_1, f_2, f_3)$ na grupi $(\mathbb{Z}_{11}^*, *)$ s $h = 5$ in $g = 2$.

Modifikacija Pollardove iteracijske funkcije f_P

Naj bosta g in h elementa končne neprazne množice G. Iščemo diskretni logaritem elementa h v osnovi g. Naj bosta $m, n \in \{0, \ldots, |G|-1\}$ poljubni števili. Definirajmo $M = g^m$ in $N = h^n$. Če uporabimo v definiciji iteracijsko funkcijo (5.9) neko potenco elementov g in h, dobimo **modificirano Pollardovo iteracijsko funkcijo**, ki jo označimo s f_{Pm} in definiramo s

$$f_{Pm}(y) = \begin{cases} N\,y, & y \in S_1, \\ y^2, & y \in S_2, \\ M\,y, & y \in S_3. \end{cases} \tag{5.11}$$

Zaporedji pozitivnih celih števil $\{\alpha_i\}$ in $\{\beta_i\}$ računamo po modulu $|G|$ in po pravilu

$$\alpha_{i+1} = \begin{cases} \alpha_i + n & y_i \in S_1 \\ 2\alpha_i & y_i \in S_2 \\ \alpha_i & y_i \in S_3 \end{cases} \quad \text{in} \quad \beta_{i+1} = \begin{cases} \beta_i & y_i \in S_1 \\ 2\beta_i & y_i \in S_2 \\ \beta_i + m & y_i \in S_3. \end{cases} \tag{5.12}$$

Iteracijske funkcije z $r \geq 3$ množitelji

Naj bo $r \geq 3$, $m_1, n_1, \ldots, m_r, n_r \in \{0, \ldots, |G|-1\}$ poljubna števila (ne nujno paroma različna) in $S_1, S_2, \ldots, S_r$ neka delitev množice G. Vzemimo $M_i = g^{m_i} h^{n_i}$ za $i = 1, \ldots, r$. Če uporabimo v definiciji Pollardove iteracijske funkcije $f : G \to G$ na elementih množice S_i množenje z M_i, dobimo **aditivno Pollardovo iteracijsko funkcijo**, ki jo označimo s f_T in definiramo s

$$f_T(y) = \begin{cases} M_1\,y, & y \in S_1, \\ M_2\,y, & y \in S_2, \\ \vdots & \vdots \\ M_r\,y, & y \in S_r. \end{cases} \tag{5.13}$$

Elemente M_i imenujemo **množitelji** iteracijske funkcije f_T. Iteracijsko funkcijo (5.13) imenujemo aditivna iteracijska funkcija tudi zato, ker za zaporedji pozitivnih celih števil $\{\alpha_i\}$ in $\{\beta_i\}$ velja pravilo

$$\alpha_{i+1} = \begin{cases} \alpha_i + n_1, & y_i \in S_1 \\ \alpha_i + n_2, & y_i \in S_2 \\ \vdots & \vdots \\ \alpha_i + n_r, & y_i \in S_r \end{cases} \quad \text{in} \quad \beta_{i+1} = \begin{cases} \beta_i + m_1, & y_i \in S_1 \\ \beta_i + m_2, & y_i \in S_2 \\ \vdots & \vdots \\ \beta_i + m_r, & y_i \in S_r. \end{cases} \tag{5.14}$$

Elemente zaporedij števil $\{\alpha_i\}$ in $\{\beta_i\}$ računamo po modulu $|G|$. Pollardovo zaporedje $\{y_i\}$, ki ga generiramo z uporabo aditivne Pollardove iteracijske funkcije (5.13), imenujemo **(r)-Pollardovo zaporedje**.

Iteracijske funkcije z $r \geq 3$ množitelji in $q \geq 2$ kvadriranji

Naj bosta g in h elementa končne neprazne množice G. Iščemo diskretni logaritem elementa h v osnovi g. Naj bodo $m_1, n_1, \ldots, m_r, n_r \in \{0, \ldots, |G| - 1\}$ poljubna števila (ne nujno paroma različna) in $S_1, S_2, \ldots, S_{r+q}$ neka delitev množice G. Vzemimo $M_i = g^{m_i} h^{n_i}$ za $i = 1, \ldots, r$. Če uporabimo v definiciji Pollardove iteracijske funkcije $f : G \to G$ nekaj korakov množenj in nekaj korakov kvadriranj, dobimo **mešano Pollardovo iteracijsko funkcijo**, ki jo označimo s f_C in definiramo s

$$f_C(y) = \begin{cases} M_1\, y, & y \in S_1, \\ M_2\, y, & y \in S_2, \\ \vdots & \vdots \\ M_r\, y, & y \in S_r, \\ y^2 & \text{sicer.} \end{cases} \tag{5.15}$$

Elemente M_i imenujemo **množitelji** iteracijske funkcije f_C. Iteracijsko funkcijo (5.15) imenujemo mešana iteracijska funkcija tudi zato, ker za zaporedji pozitivnih celih števil $\{\alpha_i\}$ in $\{\beta_i\}$ velja pravilo

$$\alpha_{i+1} = \begin{cases} \alpha_i + n_1, & y_i \in S_1 \\ \alpha_i + n_2, & y_i \in S_2 \\ \vdots & \vdots \\ \alpha_i + n_r, & y_i \in S_r \\ 2\,\alpha_i & \text{sicer} \end{cases} \quad \text{in} \quad \beta_{i+1} = \begin{cases} \beta_i + m_1, & y_i \in S_1 \\ \beta_i + m_2, & y_i \in S_2 \\ \vdots & \vdots \\ \beta_i + m_r, & y_i \in S_r \\ 2\,\beta_i & \text{sicer.} \end{cases} \tag{5.16}$$

Zaporedji števil $\{\alpha_i\}$ in $\{\beta_i\}$ računamo po modulu $|G|$. Pollardovo zaporedje $\{y_i\}$, ki ga generiramo z uporabo mešane Pollardove iteracijske funkcije f_C iz (5.15), imenujemo **$(r + q)$-Pollardovo zaporedje**. Do konca poglavlja bomo predstavili do sedaj definirane iteracijske funkcije tudi za primer grupe na eliptični krivulji nad končnim obsegom.

Iteracijske funkcije na eliptični krivulji

Naj bo E ciklična grupa na eliptični krivulji nad praštevilskim obsegom $\mathbb{F}_p$, $p > 3$, v kateri rešujemo ECDLP z uporabo Pollardove ρ-metode. To pomeni,

da moramo izbrati neko iteracijsko funkcijo $f : E \to E$, ki generira Pollardovo zaporedje $\{Y_i\}$ po pravilu $Y_{i+1} = f(Y_i)$ za $i \geq 0$. Recimo, da iščemo diskretni logaritem točke $Q \in E$ v osnovi $P \in E$. Potem hkrati z računanjem zaporedja $\{Y_i\}$ računamo tudi taki zaporedji pozitivnih celih števil $\{\alpha_i\}$ in $\{\beta_i\}$ po modulu $|E|$, da velja $Y_i = \alpha_i\, Q + \beta_i\, P$ za $i \geq 0$. Naj bo $x(Y)$ koordinata $x \in \mathbb{F}_p$ poljubne točke $Y \in E$. Zaporedje $\{Y_i\}$ lahko generiramo z uporabo Pollardove iteracijske funkcije (4.4), ki jo na eliptični krivulji E definiramo s

$$f_P(Y) = \begin{cases} Q + Y, & 0 \leq x(Y) \leq \frac{p}{3}, \\ 2\,Y, & \frac{p}{3} < x(Y) < \frac{2p}{3}, \\ P + Y, & \frac{2p}{3} \leq x(Y) < p. \end{cases} \tag{5.17}$$

Iteracijsko funkcijo lahko definiramo tudi tako, da uporabimo delitev iz tabele 5.1 za množico točk grupe E. Naj bo S_1, S_2 in S_3 neka delitev množice točk grupe E. S to delitvijo dobimo **posplošeno Pollardovo iteracijsko funkcijo**, ki jo definiramo s

$$f_P(Y) = \begin{cases} Q + Y, & Y \in S_1, \\ 2\,Y, & Y \in S_2, \\ P + Y, & Y \in S_3. \end{cases} \tag{5.18}$$

Zaporedji pozitivnih celih števil $\{\alpha_i\}$ in $\{\beta_i\}$ računamo po modulu $|E|$ in po pravilu (5.10). Naj bosta $m, n \in \{0, \ldots |E| - 1\}$ poljubni števili in vzemimo $M = m\,P$ in $N = n\,Q$. Če uporabimo v definiciji Pollardove iteracijske funkcije $f : E \to E$ točki M in N, dobimo **modificirano Pollardovo iteracijsko funkcijo**, ki jo označimo s f_{Pm} in definiramo s

$$f_{Pm}(Y) = \begin{cases} N + Y, & Y \in S_1, \\ 2\,Y, & Y \in S_2, \\ M + Y, & Y \in S_3. \end{cases} \tag{5.19}$$

Zaporedji pozitivnih celih števil $\{\alpha_i\}$ in $\{\beta_i\}$ računamo po modulu $|E|$ in po pravilu (5.12). Naj bo $r \geq 3$, $S_1, \ldots, S_r$ neka delitev množice točk grupe E in $m_i, n_i \in \{0, \ldots, |E| - 1\}$ za $i = 1, \ldots, r$ poljubna števila (ne nujno paroma različna). Vzemimo $M_i = m_i\,P + n_i\,Q$ za $i = 1, \ldots, r$. Če uporabimo v definiciji Pollardove iteracijske funkcije $f : E \to E$ točke M_i, dobimo **aditivno**

Pollardovo iteracijsko funkcijo, ki jo označimo s f_T in definiramo s

$$f_T(Y) = \begin{cases} M_1 + Y, & Y \in S_1, \\ M_2 + Y, & Y \in S_2, \\ \vdots & \vdots \\ M_r + Y, & Y \in S_r. \end{cases} \tag{5.20}$$

Točke M_i imenujemo **aditivne konstante** iteracijske funkcije f_T. Zaporedji pozitivnih celih števil $\{\alpha_i\}$ in $\{\beta_i\}$ računamo po modulu $|E|$ in po pravilu (5.14). Zaporedje $\{Y_i\}$, ki ga generiramo z uporabo aditivne Pollardove iteracijske funkcije (5.20), imenujemo **(r)-Pollardovo zaporedje**. Naj bo $r \geq 3$, $q \geq 2$, $S_1, S_2, \ldots, S_{r+q}$ neka delitev množice točk grupe E in $m_1, n_1, \ldots, m_r, n_r \in \{0, \ldots, |E| - 1\}$ poljubna števila (ne nujno paroma različna). Vzemimo $M_i = m_i\, P + n_i\, Q$ za $i = 1, \ldots, r$. Če uporabimo v definiciji Pollardove iteracijske funkcije $f : E \to E$ točke M_i, dobimo **mešano Pollardovo iteracijsko funkcijo**, ki jo označimo s f_C in definiramo s

$$f_C(Y) = \begin{cases} M_1 + Y, & Y \in S_1, \\ M_2 + Y, & Y \in S_2, \\ \vdots & \vdots \\ M_r + Y, & Y \in S_r, \\ 2\,Y & \text{sicer.} \end{cases} \tag{5.21}$$

Točke M_i imenujemo **aditivne konstante** iteracijske funkcije f_C. Zaporedji pozitivnih celih števil $\{\alpha_i\}$ in $\{\beta_i\}$ računamo po modulu $|E|$ in po pravilu (5.16). Zaporedje $\{Y_i\}$, ki ga generiramo z uporabo mešane Pollardove iteracijske funkcije (5.21), imenujemo **$(r + q)$-Pollardovo zaporedje**.

Poglavje 6

ECDLP v praksi

V tem poglavju si bomo ogledali reševanje ECDLP v praksi. V ta namen bomo predstavili Certicomove izzive, ki jih sestavlja 24 primerov ECDLP. Večji del poglavja bomo namenili opisu zadnjih dveh rešitev. Kot že vemo, je Pollardova ρ-metoda trenutno najboljši algoritem za reševanje ECDLP.

6.1 Predstavitev

Kanadsko podjetje Certicom Corp. je 6. novembra 1997 na svoji domači strani postavilo primere ECDLP v izbranih grupah na eliptični krivulji nad končnim obsegom v javno reševanje [3]. Reševanje teh izzivov je motivirano tudi z denarnimi nagradami. V tem razdelku jih bomo na kratko predstavili.

Glede na izbiro eliptične krivulje in končnega obsega delimo Certicomove izzive na tri razrede. V prvi razred so postavljeni primeri ECDLP nad obsegi lihe karakteristike, v drugi razred nad obsegi karakteristike 2, v tretji razred pa na Koblitzovih krivuljah. Njihove oznake so: ECC$p-k$, ECC$2-k$ in ECC$2K-k$. Oznaka p nam pove, da je ECDLP postavljen nad obsegom $\mathbb{F}_p$, oznaka K pa, da je definiran na Koblitzovi krivulji. Število k določa velikost grupe na eliptični krivulji, v kateri je izziv postavljen. Koblitzove krivulje oziroma **anomalne krivulje** imenujemo razred eliptičnih krivulj nad obsegi s karakteristiko 2 oblike $y^2 + xy = x^3 + ax^2 + b$, kjer je $a, b \in \{0, 1\}$. Vsi Certicomovi izzivi so (rešeni izzivi so označeni poudarjeno):

1. **ECC$p-79$**, **ECC$p-89$**, **ECC$p-97$**, **ECC$p-109$**, ECC$p-131$, ECC$p-163$, ECC$p-191$, ECC$p-239$, ECC$p-359$,

Certicomov izziv	datum rešitve	število operacij na E do rešitve	število iteracij na sekundo	število dni enega računalnika
ECC$p-79$	6. dec. 1997	$1,4 \cdot 10^{12}$	314000	52
ECC2 $-$ 79	16. dec. 1997	$1,7 \cdot 10^{12}$	170000	116
ECC$p-89$	12. jan. 1998	$2,4 \cdot 10^{13}$	388000	716
ECC2 $-$ 89	9. feb. 1998	$1,8 \cdot 10^{13}$	187000	1114
ECC$p-97$	18. mar. 1998	$2,0 \cdot 10^{14}$	361000	6412
ECC2$K-95$	21. maj 1998	$2,2 \cdot 10^{13}$	149000	1709
ECC2 $-$ 97	22. sept. 1999	$1,2 \cdot 10^{14}$	227000	6118
ECC2$K-108$	**4. april 2000**	$\mathbf{2,3 \cdot 10^{15}}$	**160364**	**166000**
ECC$p-109$	**15. okt. 2002**	$\mathbf{3,6 \cdot 10^{16}}$	**351000**	**1187100**

Tabela 6.1: Rešeni Ceticomovi izzivi. V tretjem stolpcu imamo število opravljenih operacij v grupi na eliptični krivulji do rešitve, v četrtem in petem stolpcu pa imamo število narejenih operacij na sekundo ter število dni računanja na računalniku Digital Alpha (500MHz) z operacijskim sistemom Linux. Za rešitev izziva ECC2$K-108$ bi potrebovali z enim takim računalnikom približno $455(\doteq 166000/365)$ let. Za rešitev izziva ECC$p-109$ pa bi potrebovali približno $3250(\doteq 1187100/365)$ let po Certicomovih ocenah, glej §6.2.

2. **ECC2 $-$ 79**, **ECC2 $-$ 89**, **ECC2 $-$ 97**, ECC2 $-$ 109, ECC2 $-$ 131, ECC2 $-$ 163, ECC2 $-$ 191, ECC2 $-$ 238, ECC2 $-$ 353,
3. **ECC2$K-95$**, **ECC2$K-108$**, ECC2$K-130$, ECC2$K-163$, ECC2$K-238$ in ECC2$K-358$.

Naj bo E oznaka grupe na eliptični krivulji nad končnim obsegom. Specifikacije do danes rešenih Certicomovih izzivov podamo v tabeli 6.1, ki je povzeta iz njihove domače strani. Kot smo že povedali, so bile vse do danes prejete rešitve Certicomovih izzivov izračunane z uporabo Pollardove ρ-metode. Zato lahko rezultate Certicomovih izzivov uporabimo tudi za potrjevanje teoretično izpeljane zahtevnosti Pollardove ρ-metode, glej §3.3. Slednje podamo v tabeli 6.2. Pri tem predpostavimo, da so bila generiranja Pollardovih zaporedij v vseh implementacijah Pollardove ρ-metode za reševanje Certicomovih izzivov naključna. Če primerjamo četrti in peti stolpec te tabele hitro ugotovimo, da se teoretična

Certicomov izziv	obseg (biti)	grupa (biti, m)	teorija $\sqrt{2^m}$	št. operacij na E do rešitve, j	k $j = k\sqrt{2^m}$
ECC$p-79$	79	79	$2^{39,5}$	$2^{40,3}$	$1,741$
ECC2-79	79	79	$2^{39,5}$	$2^{40,6}$	$2,143$
ECC$p-89$	89	89	$2^{44,5}$	$2^{44,4}$	$0,933$
ECC2-89	89	89	$2^{44,5}$	$2^{44,0}$	$0,707$
ECC$p-97$	97	97	$2^{48,5}$	$2^{47,5}$	$0,500$
ECC2$K-95$	97	95	$2^{47,5}$	$2^{44,3}$	$0,109$
ECC2-97	97	97	$2^{48,5}$	$2^{46,8}$	$0,308$
ECC2$K-108$	**109**	**108**	$\mathbf{2^{54,0}}$	$\mathbf{2^{51,0}}$	$\mathbf{0,125}$
ECC$p-109$	**109**	**109**	$\mathbf{2^{54,5}}$	$\mathbf{2^{55,0}}$	$\mathbf{1,414}$

Tabela 6.2: Rešeni Certicomovi izzivi potrjujejo teorijo.

zahtevnost Pollardove ρ-metode zelo ujema s prakso, glej šesti stolpec tabele 6.2. Pripomnimo, en korak Pollardove ρ-metode za reševanje ECDLP pomeni eno seštevanje na eliptični krivulji, če uporabimo Brentov algoritem ali pa algoritem primerjaj in uredi za iskanje trčenja v Pollardovem zaporedju. Zato pove število korakov do rešitve tudi število opravljenih seštevanj na eliptični krivulji.

6.2 ECC2$K-108$ in ECC$p-109$

Izziv **ECC2$K-108$** je bil postavljen na 108-bitni podgrupi E_1 grupe E, ki je bila definirana na množici točk Koblitzove krivulje

$$y^2 + xy = x^3 + x^2 + 1$$

nad obsegom $\mathbb{F}_{2^{109}}$, tj. nad 109-bitnim obsegom. Grupa E je reda $2\,p$, kjer je

$$p = 324\,518\,553\,658\,426\,701\,487\,448\,656\,461\,467$$

108-bitno praštevilo. Na Certicomu so izbrali za generator 108-bitne praštevilske grupe E_1 točko

$$P = (90\,697\,808\,493\,257\,888\,388\,328\,391\,275\,095,\\ 618\,090\,430\,167\,842\,045\,420\,788\,088\,541\,417)$$

in nato še zelo veliko število x ter izračunali točko $Q = x\,P$, tj.

$$Q = (647\,833\,325\,203\,157\,411\,092\,704\,311\,887\,262,\\ 631\,362\,321\,770\,629\,145\,327\,378\,661\,647\,389).$$

Na koncu so postavili izziv: za dani točki P in Q iz grupe E_1, kjer je P generator grupe E_1, najdi najmanjše tako število

$$x \in \{0, \ldots, 324\,518\,553\,658\,426\,701\,487\,448\,656\,461\,466\},$$

da bo veljalo $Q = x\,P$ v grupi E_1. Reševanja izziva se je lotila ekipa pod vodstvom Roberta Harleya, ki je rešitev tudi uspešno našla. To se je zgodilo 4. aprila 2000. Rešitev izziva ECC2$K-108$ je

$$\log_P Q = 47\,455\,661\,896\,223\,045\,299\,748\,316\,018\,941.$$

Na Certicomu so hkrati s postavitvijo izzivov predstavili tudi oceno potrebnega časa za njihovo reševanje z uporabo Pollardove ρ-metode. Certicomova ocena potrebnega časa za izziv ECC2$K-108$ je bila $1,3 \cdot 10^6$ dni računanja na računalniku Pentium (100MHz), tj. približno 3.560 let. Reševanje izziva ECC2$K-108$ pa je dejansko trajalo štiri mesece in je vključevalo približno 9.500 računalnikov, med katerimi so bili najbolj zastopani računalniki Digital Alpha (500MHz) z operacijskim sistemom Linux. Pri reševanju je sodelovalo okrog 1300 ljudi. Tak čas reševanja izziva ECC2$K-108$ ni nobeno presenečenje glede na Certicomovo oceno, glej tabelo 6.2. Pripomnimo, da je reševanje izziva ECC2$K-108$ zahtevalo približno 50-krat več dela kot ga je bilo potrebno vložiti za razbitje izziva RSA-512, tj. za razcep 512-bitnega oziroma 155-mestnega RSA ključa. Izziv RSA-512 je bil rešen 22. avgusta 1999, za podrobnosti glej naslov: (`http://www.rsasecurity.com/rsalabs/challenges/`).

Izziv **ECC$p-109$** je bil postavljen na 109-bitni praštevilski grupi E, podani z enačbo

$$y^2 = x^3 + 321\,094\,768\,129\,147\,601\,892\,514\,872\,825\,668\,x + \\ 430\,782\,315\,140\,218\,274\,262\,276\,694\,323\,197$$

nad obsegom $\mathbb{F}_p$, kjer je $p = 564\,538\,252\,084\,441\,556\,247\,016\,902\,735\,257$. Število p je 109-bitno praštevilo. Pripomnimo, grupa E ima

$$564\,538\,252\,084\,441\,531\,840\,258\,143\,378\,149$$

elementov. Slednje število je prav tako 109-bitno praštevilo. Na Certicomu so izbrali za generator 109-bitne praštevilske grupe E točko

$$\begin{aligned} P = (&97\,339\,010\,987\,059\,066\,523\,156\,133\,908\,935,\\ &149\,670\,372\,846\,169\,285\,760\,682\,371\,978\,898) \end{aligned}$$

in nato še zelo veliko število x ter izračunali točko $Q = x\,P$, tj.

$$\begin{aligned} Q = (&44\,646\,769\,697\,405\,861\,057\,630\,861\,884\,284,\\ &522\,968\,098\,895\,785\,888\,047\,540\,374\,779\,097). \end{aligned}$$

Na koncu so postavili izziv: za dani točki P in Q iz grupe E, kjer je P generator grupe E, najdi najmanjše tako število

$$x \in \{0, \ldots, 564\,538\,252\,084\,441\,531\,840\,258\,143\,378\,148\},$$

da bo veljalo $Q = x\,P$ v grupi E. Reševanja izziva se je lotila ekipa pod vodstvom Chrisa Monica 14. aprila 2001, za podrobnosti glej naslov: (`http://www.nd.edu/~cmonico/eccp109/`). Na Certicomu so, kot smo že povedali, s postavitvijo izzivov predstavili tudi oceno potrebnega časa za reševanje le-teh z uporabo Pollardove ρ-metode. Certicomova ocena potrebnega časa za najdenje rešitve izziva ECC$p-109$ je $9,0 \cdot 10^6$ dni računanja na računalniku Pentium (100MHz), tj. približno 25.000 let, kar je ocena za približno 7-krat več dela kot ga je bilo potrebnega za rešitev izziva ECC2$K-108$. Glede na čas reševanja izziva ECC2$K-108$ in oceno potrebnega časa za najdenje rešitve izziva ECC$p-109$ je bilo pričakovati rešitev slednjega izziva po približno 28-ih mesecih računanja. Vendar se je le-to zgodilo po 18-tih mesecih računanja, kar pa ni nič presenetljivega. Po tolikem času je bila namreč ocena za verjetnost dogodka vsaj enega trčenja v Pollardovi ρ-metodi že $0,64$. Izziv ECC$p-109$ je bil rešen 15. oktobra 2002, kar so tudi uradno potrdili na Certicomovi domači strani. Rešitev izziva ECC$p-109$ je

$$\log_P Q = 281\,183\,840\,311\,601\,949\,668\,207\,954\,530\,684.$$

Poglavje 7

Zaključek

V tem delu so na kratko predstavljeni naslednji algoritmi za reševanje DLP: metoda grobe sile (uvod), razširjen Evklidov algoritem (§1.2), metoda veliki-mali korak (§2.1), metoda index-calculus (§2.2) in Pohlig-Hellmanov algoritem (§2.3). Natančno pa sta predstavljeni Pollardova ρ-metoda v §3 in Pollard-Floydova ρ-metoda v §4. V tabeli 7.1 so podane časovne zahtevnosti nekaterih algoritmov za reševanje DLP v poljubni praštevilski ciklični grupi G.

	$\lvert G\rvert = p$	$p \doteq 2^{113}$	$p \doteq 2^{160}$	$p \doteq 2^{190}$	$p \doteq 2^{256}$
metoda grobe sile	$O(p)$	2^{113}	2^{160}	2^{190}	2^{256}
metoda veliki-mali korak	$O(\sqrt{p}\,)$	$2^{56,5}$	2^{80}	2^{95}	2^{128}
Pollardova ρ-metoda	$O(\sqrt{p}\,)$	$2^{56,5}$	2^{80}	2^{95}	2^{128}
Pohlig-Hellmanov algoritem	$O(\sqrt{p}\,)$	$2^{56,5}$	2^{80}	2^{95}	2^{128}
metoda index calculus	$L_p[1/3,\, 1,923]$	2^{32}	2^{37}	2^{41}	2^{47}

Tabela 7.1: Primerjava ocen časovnih zahtevnosti algoritmov za reševanje DLP v poljubni ciklični grupi G reda p, kjer je p praštevilo, in v poljubnih 113, 160, 190 in 256-bitnih praštevilskih grupah G.

Funkcija L_p, ki je uporabljena v tabeli 7.1, je standardna oznaka za podekspo-

nentno zahtevnost. Definirana je z

$$L_p[\alpha, c] = O\left(e^{(c+o(1))(\ln n)^{\alpha}(\ln\ln n)^{1-\alpha}}\right),$$

kjer je $c \geq 0$ in $0 < \alpha < 1$, glej Menezes et al. [23, (2.3)]. Za $\alpha = 0$ je $L_p[0, c]$ polinomska funkcija v $\ln p$, za $\alpha = 1$ pa je $L_p[1, c]$ polinomska funkcija v p in zato eksponentna v $\ln p$. Časovno zahtevnost $L_p[1/3,\ 1,923]$ metode index calculus za reševanje DLP v multiplikativni grupi praštevilskega obsega $\mathbb{F}_p$ je pokazal Gordon [10]. Izmed algoritmov za reševanje DLP, ki jih nismo predstavili v tem delu, bi izpostavili Pollardovo λ-metodo, glej Pollard [26]. Njena časovna zahtevnost je $O(\sqrt{b-a})$ grupnih operacij, če vemo, da leži diskretni logaritem na intervalu $[a, b]$, prostorska zahtevnost pa je konstantna. Algoritme za reševanje DLP, katerih časovna zahtevnost je $O(\sqrt{n}\,)$ grupnih operacij, imenujemo tudi **korenski** algoritmi, glej Menezes et al. [22, str. 123]. Algoritmi za reševanje DLP, ki ne upoštevajo nobene lastnosti predstavitve grupnih elementov, delujejo v poljubni končni ciklični podgrupi. Zato jih imenujemo tudi **osnovni** algoritmi ali algoritmi s **črno škatlo** (angl. generic ali black box algorithm), glej Stinson [30, str. 239]. Naj bo $\langle g\rangle$ končna ciklična podgrupa, glej (1). Za osnovni algoritem potrebujemo naslednji dejstvi

- enoličnost zapisa vsakega elementa ciklične podgrupe v obliki zaporedja ničel in enk, tj. v binarni obliki;
- orakelj v črni škatli (angl. black box oracle), ki učinkovito izračuna kompozitum in inverz, tj.
  1. za vsak par elementov $g_1, g_2 \in \langle g\rangle$ izračuna $g_1 \circ g_2 \in \langle g\rangle$;
  2. za vsak element $g_1 \in \langle g\rangle$ izračuna $g_1^{-1} \in \langle g\rangle$.

Naj bo e enota ciklične podgrupe $\langle g\rangle$. V osnovnih algoritmih lahko na osnovi zgoraj napisanih dejstev iz vsakega elementa grupe določimo njeno enoto, tj. $e = g_1 \circ g_1^{-1}$ za vsak $g_1 \in \langle g\rangle$. Enoličen zapis elementov grupe pomeni enostavno preverjanje relacije enakosti. Torej lahko za vsak par elementov $g_1, g_2 \in \langle g\rangle$ hitro ugotovimo, ali velja $g_1 = g_2$. Kot smo videli, se operacija primerjanja enakosti grupnih elementov v algoritmih za reševanje DLP zelo pogosto uporablja. Poleg metode grobe sile, ki je tipičen primer osnovnega algoritma za reševanje DLP, so to tudi korenski algoritmi. Metoda index calculus pa je predstavnik razreda algoritmov za reševanje DLP, ki ni osnovni algoritem. Med slednje algoritme

za reševanje DLP lahko štejemo tudi razširjen Evklidov algoritem, glej §1.2. Algoritme ločimo tudi po tem, ali so deterministični ali ne. Nedeterministični algoritmi so običajno verjetnostni (angl. randomized algorithms), glej Menezes et al. [23, §2.3.4]. Verjetnostni algoritmi uporabljajo pri svojem izvajanju naključna števila. Z njimi dosegamo v kriptografiji pogosto boljše rezultate kot z determinističnimi. Glede na to delitev sta metoda grobe sile in metoda veliki-mali korak primera determinističnega algoritma za reševanje DLP. Pollardovi ρ in λ-metodi, Pohlig-Hellmanov algoritem in metoda index calculus pa so primeri verjetnostnih algoritmov za reševanje DLP.

V kriptografiji je pomembno, da izberemo za gradnjo kriptosistema tako končno ciklično podgrupo, za katero je najboljši algoritem za reševanje DLP eksponentne časovne zahtevnosti. Zato je dobro vedeti, v katerih grupah lahko uporabimo znane algoritme za reševanje DLP. Glede na to delimo znane algoritme za reševanje DLP na:

- algoritme, ki delujejo v poljubni končni ciklični podgrupi, npr. metoda grobe sile, metoda veliki-mali korak in Pollardovi ρ in λ-metodi;
- algoritme, ki delujejo v poljubni končni ciklični podgrupi in upoštevajo strukturo dane grupe, npr. Pohlig-Hellmanov algoritem;
- metode index calculus;
- metode, ki uporabijo znan izomorfizem končne ciklične podgrupe,

glej Menezes et al [22, str. 122]. Za varno uporabo težavnosti problema diskretnega logaritma v kriptografske namene je odločilno, da nimamo kakega podeksponentnega oziroma polinomskega algoritma za njegovo računanje, ki bi ga lahko uporabili v poljubni končni ciklični podgrupi.

Kot smo videli, je težavnost reševanja DLP vključena tako v varnost DSA kriptositema kot tudi v varnost RSA kriptosistema (glej §1.2). Uporabniki kriptosistemov, katerih varnost temelji na težavnosti reševanja DLP, moramo neprestano spremljati razvoj algoritmov za računanje diskretnega logaritma. Po drugi strani pa lahko vedno izberemo tako grupo, za katero se vsaj domneva, da kakega učinkovitega algoritma za reševanje DLP v izbrani grupi po vsej verjetnosti ne bomo nikoli našli. To dejstvo je podkrepljeno z naslednjim rezultatom. V. Shoup je pokazal, da je $\Omega(\sqrt{p}\,)$ pri določenih predpostavkah pričakovana asimp-

totična časovna zahtevnost najboljših algoritmov za reševanje DLP v poljubni končni grupi G, kjer je p največje praštevilo, ki deli red grupe G [28].

En razred takih algoritmov, ki dosežejo $\Omega(\sqrt{p})$ pričakovano asimptotično časovno zahtevnost, predstavljajo korenski algoritmi. Domneva se, da so splošne grupe na eliptični krivulji nad končnim obsegom predstavnik takega razreda grup, v katerih so najboljši algoritmi za reševanje DLP korenski algoritmi. Nepoznavanje metode index calculus za splošno grupo na eliptični krivulji nad končnim obsegom je za kriptografijo pozitivno. Ker so korenski algoritmi za računanje diskretnega logaritma v omenjeni grupi najboljši, je dovolj, da za gradnjo kriptosistema izberemo 160-bitno praštevilsko podgrupo grupe na eliptični krivulji nad končnim obsegom. Kot smo pokazali v §4, ima Pollardova ρ-metoda med korenskimi algoritmi veliko prednost zaradi njene konstantne prostorske zahtevnosti. Za zaključek izpostavimo odprt problem: najdi index calculus metodo za reševanje ECDLP. Še bolj pa je aktualen naslednji odprt problem: najdi index calculus metodo za reševanje ECDLP na Koblitzovi krivulji, glej str. 95.

Dodatek A

Zahtevnost algoritma

Algoritme najpogosteje ločujemo med seboj glede na porabljen čas in prostor. Ker je natančen čas in prostor težko opredeliti, si pomagamo z **asimptotičnimi ocenami**, ki nam povedo, kako se potreben čas in prostor povečujeta s povečanjem vhodnih podatkov. Te asimptotične ocene imenujemo **časovna** in **prostorska zahtevnost**. Za predstavitev zahtevnosti uporabljamo ***O*-notacijo**, ***o*-notacijo** in **Ω-notacijo**. Le-te bomo sedaj definirali, glej Menezes et al. [23, §2.3.2]. Naj bosta $f, g : \mathbb{N} \to \mathbb{R}$ neki funkciji. Pravimo, da

- f **asimptotično ne raste hitreje** kot g oziroma, da je g **asimptotična zgornja meja** za f (angl. asymptotic upper bound), če obstajata taki konstanti $c > 0$ in $n_0 > 0$, da velja $0 \leq f(n) \leq c\,g(n)$ za vse $n \geq n_0$, ali ekvivalentno, če obstaja $\lim_{n\to+\infty} f(n)/g(n)$. Simbolično bomo to lastnost zapisali z $f(n) = O(g(n))$ ali krajše z $f = O(g)$. Na primer, $n^2 + 3n + 7 = O(n^2)$;
- f **narašča asimptotično počasneje** kot g, če za vsako konstanto $c > 0$ obstaja tako pozitivno število n_0, da velja $0 \leq f(n) < c\,g(n)$ za vse $n \geq n_0$, ali ekvivalentno, če obstaja $\lim_{n\to+\infty} f(n)/g(n)$ in je enaka 0. Simbolično bomo to lastnost zapisali z $f(n) = o(g(n))$ ali krajše z $f = o(g)$. Na primer, $n^2 = o(n^5)$.
- f **narašča asimptotično vsaj tako hitro** kot g oziroma, da je g **asimptotična spodnja meja** za f (angl. asymptotic lower bound), če obstajata taki konstanti $c > 0$ in $n_0 > 0$, da velja $0 \leq c\,g(n) \leq f(n)$ za vse $n \geq n_0$. Simbolično bomo to lastnost zapisali z $f(n) = \Omega(g(n))$ ali krajše

z $f = \Omega(g)$. Velja $f = \Omega(g)$ natanko tedaj, ko je $g = O(f)$. Torej je npr. $n^2 + 3n + 7 = \Omega(n^2)$.

Velikost vhodnih podatkov v algoritem se meri s številom bitov, ki jih zasedajo. Na primer, za zapis števila n potrebujemo $m = \lfloor \log_2 n \rfloor + 1$ bitov. Zato običajno izrazimo zahtevnost algoritma v odvisnosti od števila m. S tem dosežemo najbolj jasno predstavo o zahtevnosti algoritma. Naj bo f časovna zahtevnost algoritma in e osnova naravnega logaritma. Glede na funkcijo f delimo algoritme med drugim na

- **konstantne**, tj. $f(m) = O(1)$. Na primer, $f(m) = c$, kjer je c poljubna pozitivna konstanta;
- **polinomske**, tj. $f(m) = O(m^k)$, kjer je k neka pozitivna konstanta. Na primer, $f(m) = m^4$;
- **podeksponentne**, tj. $f(m) = e^{g(m)}$, kjer je $g : \mathbb{N} \to \mathbb{R}$ taka funkcija, za katero velja $g(m) = o(m)$. Na primer, $f(m) = e^{m^a}$ za $0 < a < 1$;
- **eksponentne**, tj. $f(m) = e^{g(m)}$, kjer je $g : \mathbb{N} \to \mathbb{R}$ taka funkcija, za katero velja $g(m) = O(m^k)$, in k neka pozitivna konstanta. Na primer, $f(m) = e^m$.

Polinomske algoritme običajno smatramo za **učinkovite**, eksponentne pa za **neučinkovite**. Podeksponentni algoritmi pa so tisti, katerih časovna zahtevnost je med polinomsko in eksponentno zahtevnostjo. Ker delamo v končnih grupah, lahko privzamemo, da opravimo vsako grupno operacijo v konstantnem času in da potrebujemo za zapis vsakega elementa iz dane grupe konstanten prostor, tj. neko določeno število bitov. Zato običajno merimo časovno zahtevnost vsakega algoritma za reševanje DLP s številom opravljenih grupnih operacij, prostorsko zahtevnost pa s številom shranjenih grupnih elementov.

Literatura

[1] J. Barbič, *Schoofov algoritem*, diplomsko delo, Fakulteta za matematiko in fiziko, Univerza v Ljubljani, 2000, (`http://valjhun.fmf.uni-lj.si/~ajurisic/teaching.html#mentor`).

[2] R. P. Brent, *An improved Monte Carlo factorization algorithm*, BIT **20** (1980), 176-184.

[3] Certicom Corp., (`http://www.certicom.ca/resources/ecc_chall/`).

[4] C. C. Cocks, *A note on non-secret encryption*, CESG Research Report, 1973, (`http://www.cesg.gov.uk/publications/media/nsecret/notense.pdf`).

[5] H. Cohen, *A Course in Computational Algebraic Number Theory, Fourth Printing 2000*, Springer-Verlag, Berlin Heidelberg, 1993.

[6] D. Coppersmith, A. M. Odlyzko in R. Schroeppel, *Discrete logarithm in* GF(p), Algorithmica **1** (1986), 1-15.

[7] W. Diffie in M. E. Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory **22** (1976) ,644-654.

[8] T. ElGamal, *A public key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Transactions on Information Theory **31** (1985), 469-472.

[9] J. H. Ellis, *The possibility of non-secret encryption*, CESG Research Report, 1970, (`http://www.cesg.gov.uk/publications/media/nsecret/possnse.pdf`).

[10] D. Gordon, *Discrete logarithms in* $\mathbb{F}_p$ *using number field sieve*, SIAM Journal on Discrete Mathematics **6** (1993), 124-138.

[11] A. Jurišić in A. J. Menezes, *Elliptic Curves and Cryptography*, Dr. Dobb's Journal **264** (1997), 26-36.

[12] A. Jurišić, *Tečaj iz kriptografije*, (`http://valjhun.fmf.uni-lj.si/~ajurisic/tecaj2/`).

[13] A. Jurišić in J. Tonejc, *Pametne kartice in varnost - 1. del*, Monitor **6** (2001), 66-75.

[14] A. Jurišić in J. Tonejc, *Pametne kartice - 2. del, Zasebno življenje javnih ključev*, Monitor **7/8** (2001), 44-51.

[15] A. Jurišić in J. Tonejc, *Pametne kartice - 3. del, Napadi in obrambe: velike skrivnosti malih kartic*, Monitor **9** (2001), 54-64.

[16] M. Juvan, *O Evklidovem algoritmu*, Presek **21**/2 (1993-94), 116-121.

[17] D. Knuth, *The art of computer programming, Volume 2: Seminumerical algorithms*, Reading, Massachusetts, Addison-Wesley, 1969.

[18] D. Knuth, *The art of computer programming, Volume 3: Sorting and Searching*, Reading, Massachusetts, Addison-Wesley, 1973.

[19] N. Koblitz, *Elliptic curve cryptosystems*, Mathematics of Computation **48** (1987), 203-209.

[20] U. M. Maurer in S. Wolf, *On the Complexity of Breaking the Diffie-Hellman Protocol*, (`http://www.crypto.ethz.ch/publications.html`).

[21] K. S. McCurley, *The discrete logarithm problem*, Cryptology and Computational Number Theory **42** (1990), 49-74.

[22] A. J. Menezes, I. F. Blake, X. Gao, R. C. Mullin, S. A. Vanstone in T. Yaghoobian, *Applications of Finite Fields*, Kluwer Academic Publishers, 1993.

[23] A. J. Menezes, P. C. Van Oorschot in S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[24] V. Miller, *Uses of elliptic curves in cryptography*, Advances in Cryptology-CRYPTO'85 Proceedings, LNCS **218** (1986), 417-426.

[25] R. Petek, *RSA kriptosistem*, diplomsko delo, Fakulteta za matematiko in fiziko, Univerza v Ljubljani, 2000, (`http://valjhun.fmf.uni-lj.si/~ajurisic/teaching.html#mentor`).

[26] J. M. Pollard, *Monte Carlo methods for index computation* (mod p), Mathematics of Computation **32** (1978), 918-924.

[27] M. Razpet, *Zlati pravokotnik*, Presek **27**/1 (1999-2000), 34-43.

[28] V. Shoup, *Lower Bounds for Discrete Logarithms and Related Problems*, Advances in Cryptology-EUROCRYPT'97 Proceedings, LNCS **1233** (1997), 256-266.

[29] S. Singh, *The Code Book*, Anchor Books, 1999.

[30] D. R. Stinson, *Cryptography: Theory and Practice, Second Edition*, CRC Press, 2002.

[31] E. E. Teske, *New Algorithms for Finite Abelian Groups*, Ph.D. Thesis, Technische Univerität Darmstadt, 1998.

[32] I. Vidav, *Algebra, 4. natis*, Društvo matematikov, fizikov in astronomov Slovenije, 1989.

[33] I. Vidav, *Eliptične krivulje in eliptične funkcije*, Društvo matematikov, fizikov in astronomov Slovenije, 1991.