ŠOLSKI CENTER CELJE

Srednja šola za kemijo, elektrotehniko in računalništvo

KRIPTOGRAFIJA Z ELIPTIČNIMI KRIVULJAMI

raziskovalna naloga

Avtorji:

Maj Zabukovnik, R-3. b

Aney Vaishnav, R-3. b

Lan Stokavnik, R-3. b

Mentor:

Boštjan Lubej, dipl. inž. inf. in tehn. kom.

Martina Omerzel, prof. mat. in fiz.

Mestna občina Celje, Mladi za Celje

Celje, marec 2023

ŠOLSKI CENTER CELJE

Srednja šola za kemijo, elektrotehniko in računalništvo

KRIPTOGRAFIJA Z ELIPTIČNIMI KRIVULJAMI

raziskovalna naloga

Avtorji:

Maj Zabukovnik, R-3. b

Aney Vaishnav, R-3. b

Lan Stokavnik, R-3. b

Mentor:

Boštjan Lubej, dipl. inž. inf. in tehn. kom.

Martina Omerzel, prof. mat. in fiz.

Mestna občina Celje, Mladi za Celje

Celje, marec 2023

IZJAVA

Mentorja Boštjan Lubej in Martina Omerzel v skladu z 20. členom Pravilnika o organizaciji mladinske raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljava, da je v raziskovalni nalogi z naslovom Kriptografija z eliptičnimi krivuljami, katere avtorji so Maj Zabukovnik, Aney Vaishnav in Lan Stokavnik:

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 23. 3. 2023

žig šole

Podpis mentorja

Podpis odgovorne osebe

*

POJASNILO

V skladu z 20. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja (-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja (-ice) fotografskega gradiva, katerega ni avtor (-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.

Zahvala

Zahvaljujemo se vsem, ki so kakorkoli pomagali pri izdelavi raziskovalne naloge. Brez vaše pomoči raziskovalna naloge ne bi nastala, pa naj je šlo le za spodbudne besede, majhne ideje ali nasvete in predvsem kritike pri izdelavi, saj smo se iz njih naučili največ in tako je nastal še boljši končni izdelek.

Najprej bi se iskreno zahvalili profesorju Boštjanu Lubeju za motivacijo, da smo se lotili izdelave raziskovalne naloge, in za vse nasvete glede izdelave.

Velika zahvala pripada tudi profesorici Martini Omerzel ter Alešu Omerzelu, saj sta nam pomagala z razlago matematične teorije, ko smo jo potrebovali, tako da smo lahko korektno napisali teoretični del. Med izdelavo raziskovalne naloge sta se prav tako aktivno vključevala v izdelavo in nas s svojimi nasveti usmerjala v pravo smer.

Zahvalili bi se tudi profesorici Tjaši Verdev, ki je poskrbela za lektoriranje raziskovalne naloge in svetovanje pri oblikovanju besedila, ter profesorici Rosani Breznik, ki je prevedla in lektorirala angleški povzetek.

Povzetek

Uporaba učinkovitih kriptografskih algoritmov je ključnega pomena, če želimo ohraniti svetovni splet kot ga poznamo danes, saj ti algoritmi ščitijo naša gesla, osebne podatke in finančne transakcije.

Cilj raziskovalne naloge je bil raziskati delovanje kriptografije z eliptičnimi krivuljami, ene od najbolj uporabljenih asimetričnih kriptografskih algoritmov, in s pomočjo pridobljenega znanja izdelati uporabniško aplikacijo, ki v praksi prikaže uporabo pridobljenega znanja ter uporabniku omogoča šifrirano dvosmerno komunikacijo preko P2P omrežja. V nalogi so zapisane primerjave hitrosti izdelane knjižnice v primerjavi s standardno knjižnico za šifriranje izdelano s strani podjetja Microsoft. Izvedena je bila tudi raziskava potencialne odpornosti kriptografije pred napadi kvantnih računalnikov.

Naše ugotovitve kažejo, da je izdelana uporabniška aplikacija primerna za vsakdanjo rabo, uporabljena kriptografija deluje korektno, primerjava hitrosti s standardno knjižnico je pokazala, da je še vedno prostor za optimizacijo napisane kode.

Ključne besede: *kriptografija, asimetrična kriptografija, teorija grup, eliptične krivulje, P2P omrežje, kvantna odpornost.*

Abstract

The use of effective cryptographic algorithms is crucial if we want to preserve the World Wide Web as we know it today, since these algorithms protect our passwords, personal information, and financial transactions.

The objective of this research paper was to study the functioning of cryptography with elliptic curves, a commonly used asymmetrical cryptography algorithm. The outcome was the creation of a user application that demonstrates the implementation of the obtained knowledge and allows users to engage in encrypted two-way communication through a P2P network. In addition, we compared the performance of our library with a standard encryption library provided by Microsoft and evaluated the resistance of cryptography against attacks from quantum computers.

The results showed that the user application was suitable for everyday use and that the cryptography used performed as expected. The comparison with the standard library indicated room for improvement in terms of code optimization.

Keywords: *cryptography, asymmetric cryptography, group theory, elliptic curves, P2P network, quantum resistance.*

Kazalo vsebine

Kazalo slik

Kazalo tabel

Kazalo grafov

Uporabljene kratice

AES – Advanced Encryption Standard

DES – Data Encryption Standard

DSA – Digital Signature Algorithm

FTP – File Transfer Protocol

IDE – Integrated Development Environment

IP – Internet Protocol

LAN – Local Area Network

MAN – Metropolitan Area Network

RSA – Rivest-Shamir-Adleman

TCP – Transmission Control Protocol

WAN – Wide Area Network

WPF – Windows Presentation Foundation

1 UVOD

Kriptografija je veda, ki se osredotoča na matematične postopke manipulacije originalnih podatkov tako, da so ti nerazumljivi oziroma brezpomenski vsakomur, ki ne pozna ustreznega ključa, s katerim lahko le-te podatke osmisli. Ker živimo v digitalni dobi, kjer si vedno več občutljivih in osebnih informacij posredujemo preko javnih ter nezavarovanih kanalov, je pomembno, da uporabljamo učinkovite in predvsem varne kriptografske algoritme, ki zagotavljajo verodostojnost ter ščitijo poslane podatke pred javnim razkritjem.

Skozi zgodovino so se uporabljali predvsem simetrični algoritmi, kjer sta za šifriranje in dešifriranje podatkov uporabljena identična ključa. Ti algoritmi so pogosto uporabljeni še dandanes, vendar je njihova uporaba na svetovnem spletu nemogoča, saj bi uporaba le-teh zahtevala, da se dva popolna neznanca dogovorita za skupni ključ, ki bo uporabljen za šifriranje in dešifriranje podatkov. Ta izziv je v računalništvu predstavljal nekakšen sveti gral, katerega odkritje bi omogočilo internet kot ga poznamo danes. V sedemdesetih letih prejšnjega stoletja so s skupni močmi Whitfield Diffie, Martin Hellman in Ralph Merkel izumili izmenjavo ključev Diffie-Hellman-Merkel, ki je omogočila prej nepredstavljivo – način, kako se lahko dva popolna neznanca preko javnega kanala dogovorita o skupnemu ključu, ki ga lahko kasneje uporabita kot šifrirani ključ v kombinaciji z enim od simetričnih kriptografskih algoritmov.

S tem odkritjem je asimetrična kriptografija prišla iz povojev, kar je v naslednjih letih pripeljalo do odkritja RSA kriptografije, ki temelji na težavnosti faktorizacije velikih sestavljenih števil, ElGamal, ki je nadgradnja izmenjave ključev Diffie-Hellman-Merkel in omogoča še digitalne podpise, ter nenazadnje kriptografije z eliptičnimi krivuljami, ki temelji na nerešljivosti diskretnega logaritma.

V zadnjih letih kriptografija z eliptičnimi krivuljami z enormno hitrostjo pridobiva na priljubljenosti in uporabnosti, kajti v primerjavi z RSA in ElGamal ponuja veliko težji problem diskretnega logaritma, kar v praksi pomeni uporabo krajših ključev, hitrejše delovanje, manjšo poraba procesorskega časa in še bi lahko naštevali. Zaradi teh razlogov smo se odločili izdelati raziskovalno nalogo na temo kriptografije z eliptičnimi krivuljami, saj predstavlja sodoben in uporaben kriptografski algoritem, katerega prihodnost zaradi potencialne odpornosti pred kvantnimi računalniki izgleda obetavno.

1.1 Hipoteze

Cilj raziskovalne naloge je izdelava učinkovite in praktične aplikacije za šifriranje in dešifriranje poslane vsebine preko lokalnega omrežja.

V raziskovalni nalogi smo postavili naslednje hipoteze:

1. Eliptične krivulje so primernejša izbira za šifriranje kot $\mathbb{Z}_p^*$, saj je njihov problem diskretnega robustnejši in posledično odpornejši pred napadi.
2. Šifriranje z eliptičnimi krivuljami je varno pred napadom kvantnih računalnikov.
3. Izdelana aplikacija je primerna za vsakdanjo rabo.
4. Izdelana knjižnica za šifriranje je po hitrosti primerljiva s standardno Microsoftovo knjižnico za šifriranje.

1.2 Metode raziskovanja

Pri izdelavi raziskovalne naloge smo sprva naleteli na pomanjkanje primerne strokovne literature, saj je bila le-ta napisana bodisi preveč laično bodisi preveč strokovno. S podrobnejšim iskanjem in pridobivanju dodatnega strokovnega ter matematičnega znanja smo uspešno našli ustrezno literaturo. Velika večina literature je bila s spleta, saj ne obstaja veliko knjig na temo kriptografije, prav tako le-te niso na voljo v slovenskih knjižnicah. Na Fakulteti za računalništvo in informatiko v Ljubljani je bila izdelana diplomska naloga s podobnimi cilji kot naša, zato smo jo vzeli kot oporo.

1.3 Struktura raziskovalnega dela

V raziskovalni nalogi smo v prvem poglavju predstavili tematiko, hipoteze in metode dela. V drugem poglavju smo predstavili kriptografijo v splošnem, ki bralcu pomaga z razumevanjem nadaljnjega besedila. V tretjem poglavju smo se osredotočili na predstavitev potrebnega matematičnega znanja za razumevanje kriptografije z eliptičnimi krivuljami. V četrtem poglavju je razložen problem diskretnega logaritma, ki služi kot osnova mnogim asimetričnim kriptografskim algoritmom. V petem poglavju so podrobneje predstavljene eliptične krivulje, grupe eliptičnih krivulj, binarne operacije in algoritmi za kriptografijo z eliptičnimi krivuljami. V šestem poglavju smo se osredotočili na kvantne računalnike, njihovo delovanje ter kritično ocenili odpornost kriptografije z eliptičnimi krivuljami pred napadi kvantnih računalnikov. V sedmem poglavju je podrobneje predstavljeno omrežje P2P, ki v naši aplikacije služi kot osnova za izmenjavo zašifriranih sporočil. V osmem poglavju je podrobneje predstavljena

izdelana aplikacija. V devetem poglavju smo kritično ocenili in analizirali rezultate raziskovalne naloge ter potrdili oziroma ovrgli naše hipoteze.

2 KRIPTOGRAFIJA

Kriptografija je veda, na kateri temelji sodobni internet. Omogoča spletno bančništvo, varno prijavo v razna spletna mesta, zasebno sporočanje in še bi lahko naštevali. Kljub njeni pomembnosti v današnjem času je pojem bolj poznan strokovnjakom s področja računalništva in matematike, zato si oglejmo njeno definicijo. »Kriptografija je veda o matematičnih tehnikah za dosego informacijske varnosti, kot so zaupnost, celovitost podatkov, overjanje identitete in podatkov.« [1]

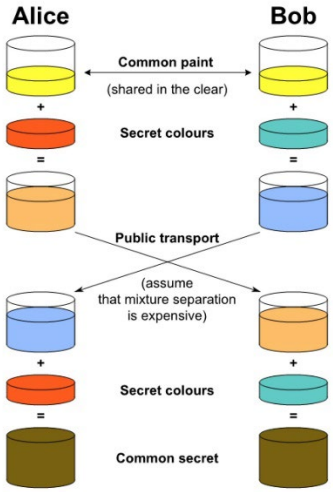
Prav zaradi široke palete uporabe obstaja več kriptografskih algoritmov. Vsak algoritem ima svoje prednosti in slabosti. Danes algoritme delimo na dve večji podskupini, in sicer simetrično ter asimetrično kriptografijo.

Zgodovinski razvoj simetričnih algoritmov se je pričel že pred približno 2000 leti. Eden od prvih simetričnih algoritmov za kriptografijo je bila Cesarjeva šifra, nato so skozi stoletja nastajale zapletenejše verzije substitucijskih šifer. Simetrična kriptografija je odigrala pomembno vlogo tudi med drugo svetovno vojno, ko so Nemci za varno sporočanje uporabljali Enigmo. Do sedemdesetih let je veljalo zmotno prepričanje, da je za varno šifriranje potrebno šifrirani algoritem skriti pred javnostjo. To zmotno mišljenje se je spremenilo, ko so leta 1977 v ZDA javno objavili DES algoritem. S to objavo se je spremenilo razmišljanje o kriptografiji, kajti varnost šifriranja ni več temeljila na nepoznavanju algoritma za šifriranje, temveč na ključu, ki je bil uporabljen za šifriranje. Čeprav je z objavo DES algoritma javnost prvič v zgodovini imela dostop do orodja, ki je omogočalo zanesljivo šifriranje, se je še vedno pojavljal problem razdeljevanja ključev, saj morata za simetrično šifriranje tako pošiljatelj kot prejemnik poznati enaka ključa.

Whitfield Diffie in Martin Hellman sta se zavedala problema razdeljevanja ključev, zato sta skušala najti rešitev. Teoretično rešitev sta našla v relativno kratkem času in jo leta 1976 objavila v raziskovalnem članku univerze v Stanfordu. Ta rešitev je bila prvi asimetrični kriptografski algoritem, ki je popolnoma odpravljal pomanjkljivost razdeljevanja ključev. V nadaljevanju je na vsakdanjem primeru predstavljena Diffie-Hellman-Merkel izmenjava ključev.

Teoretična rešitev: Ana in Bine si preko interneta želita poslati datoteko, katere vsebina mora ostati skrivnost. Vsebino bosta šifrirala s pomočjo simetričnega algoritma, vendar za to

potrebujete skupen ključ. Dogovorita se, da bo šifrirni ključ odtenek barve, ki bo nastal po končanem procesu. Ana in Bine se dogovorita, da bo rumena začetna barva. Ta podatek je lahko znan vsem, tudi Evi, ki prisluškuje in si želi pridobiti dostop do datoteke. Ana rumeni barvi primeša svojo skrivno barvo, ki je v njenem primeru rdeča. Bine stori isto, vendar je njegova skrivna barva modra. Ko sta oba zmešala začetno barvo s svojo skrivno barvo, si izmenjata dobljeni mešanici. Tudi izmenjava dobljenih mešanic je lahko javna, saj Eva iz dobljene mešanice ne more ločiti rumene barve in pa skrivne barve, ki ji je bila primešana. Prejeti mešanici nato vsak doda še svoj odtenek skrivne barve. Če sta v obeh primerih oba dodala isto količino začetne barve in isto količino skrivne barve, bi na koncu oba morala imeti enako barvo. Ta odtenek barve lahko sedaj uporabita kot ključ za šifriranje in dešifriranje poslane datoteke. Eva pa pozna le začetno barvo in obe mešanici, ki sta si jih Bine in Ana izmenjala. Iz tega pa Eva ne more razbrati skupne barve uporabljene za šifriranje. Za lažje razumevanje je ta postopek prikazan še na Sliki 1.



Slika 1: Teoretična rešitev problema Diffie-Hellman [2]

Teoretični koncept ne zveni pretirano zapleteno, vendar v matematiki do tedaj ni bila poznana ustrezna funkcija *f,* ki bi lahko generirala ista ključa, zato se je takoj po objavi pričelo iskanje primernega matematičnega algoritma, ki bi rešili problem Diffie-Hellmana. Leta 1978 sta Merkle in Hellman našla ustrezno matematično rešitev. Podrobnosti Merkle-Hellman algoritma oziroma bolj poznanega pod imenom Diffie-Hellman izmenjava ključev so predstavljene v četrtem poglavju.

Diffie-Hellman izmenjava ključev je postavila temelje za razvoj asimetrične kriptografije, kjer bi Ana in Bine lahko imela vsak svoj zasebni in javni ključ. Javni ključ bi lahko poslala vsakomur, prejemnik javnega ključa pa bi lahko sporočilo zašifriral in ga nato varno poslal preko interneta. Ko bi lastnik javnega ključa prejel zašifrirano sporočilo, bi s pomočjo svojega zasebnega ključa to sporočilo dešifriral. Kot vidimo iz zgornjega primera, se za šifriranje uporablja drugačen ključ kot za dešifriranje, zato te vrste algoritmov imenujemo asimetrični. Danes najbolj priljubljena asimetrična algoritma sta RSA in eliptične krivulje. Slednje so v nadaljevanju predstavljene podrobneje. [3]

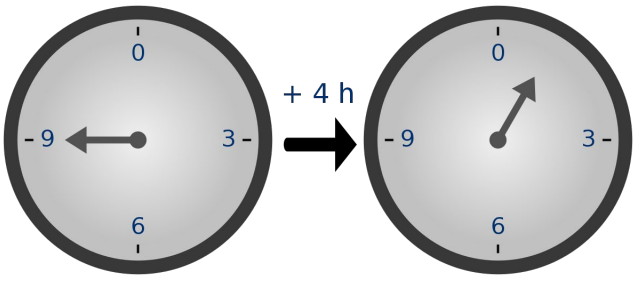
3 MATEMATIČNA TEORIJA

Asimetrični kriptografski algoritmi v veliki meri temeljijo na matematičnem znanju, zato je za njihovo razumevaje nujno potrebno nekaj matematičnega predznanja. V tretjem poglavju so zato predstavljeni modularna aritmetika, Evklidov algoritem in grupe.

3.1 Modularna aritmetika

Modularna aritmetika se uporablja v večini sodobnih asimetričnih kriptografskih algoritmov. Za lažje razumevanje si najprej oglejmo pomen vsake besede posebej. Aritmetika je del vede o matematiki, ki se ukvarja z računstvom. Večina ljudi jo uporablja vsak dan, z njo pa se je najprej srečala že v prvem razredu osnovne šole. Primer aritmetike je računanje vsote števil 3 in 5. Pridevnik modularna pa pomeni računanje, kjer se števila, ko dosežejo neko vrednost, začnejo ponavljati. Najpogostejši primer uporabe modularne aritmetike iz resničnega življenja je ura.

Primer: če je sedaj ura 9 zvečer, izračunajmo koliko bo ura čez 4 ure. Vsi se lahko strinjamo, da bo takrat ura 1 zjutraj, toda sedaj se pojavi vprašanje kako je lahko 9 + 4 = 1. Odgovor je pravilen, ker v enačbi nevede uporabimo še operacijo modul, ki je v primeru ure enak 12, saj se ura na vsakih 12 ur ponastavi na začetno vrednost. Modul predstavlja ostanek pri deljenju s tem številom. Torej 9 + 4 = 13, nato pa izračunamo izostanek 13 : 12 = 1, ostanek 1. Pravilnost rezultat pa lahko preverimo tudi s pomočjo Slike 2. Če je kazalec najprej na 9 in se nato premaknemo za 4 mesta v smeri urinega kazalca, končamo na številu 1.



Slika 2: Prikaz modularne aritmetike s pomočjo ure [4]

Modularna aritmetika se v matematiki definira s pomočjo kongruenčne relacije.

DEFINICIJA 3.1 Modularna aritmetika. Celi števili *a* in *b* sta kongruentni po modulu *m*, ki je naravno število, če in samo če m deli razliko števil *a* in *b*.

$$a, b \in \mathbb{Z}, m \in \mathbb{N},\ a \equiv b \pmod{\mathrm{m}} \Leftrightarrow m|(a-b) \quad [5]$$

Iz definicije 3.1. pa ugotovimo tudi nekaj nenavadnega – namreč obstaja neskončno mnogo ostankov, ki jih lahko zapišemo in zadostijo enačbi.

Primer: vzemimo 8 + 10 modul 12. Pravilne rešitve so torej:

- $(8 + 10) \equiv 6\ (mod\ 12)$, ker $12|((8 + 10) - 6)$,
- $(8 + 10) \equiv 18\ (mod\ 12)$, ker $12|((8 + 10) - 18)$,
- $(8 + 10) \equiv 30\ (mod\ 12)$, ker $12|((8 + 10) - 30)$ …

Kot je razvidno iz primera, se ostanki nadaljujejo v neskončnost, torej lahko zapišemo množico, katere moč ni mogoče določiti, $\{\dots -18, -6, 6, 18, 30, \dots\}$ oziroma v splošni obliki $\{\dots a - 2n, a - n, a, a + n, a + 2n, \dots\}$, kjer je *a* poljubni element iz množice ostankov in *n* vrednost modula. Takšne množice imenujemo kongruenčni razredi. Kongruenčni razredi imajo posebno lastnost, in sicer da se vsi elementi pri računanju obnašajo enako. Ta lastnost postane zelo uporabna, ko računamo z velikimi števili. Za lažje razumevanje si poglejmo primer.

Primer: recimo, da želimo izračunati $27(55 + 63)\ mod\ 11$.

Računanja se lahko lotimo po klasični aritmetiki in šele nato izračunamo modul 11, lahko pa pred seštevanjem ter množenjem z vsakim številom že izvedemo operacijo mod 11.

1. način: $27 \cdot (55 + 63) = 27 \cdot 118 = 3186 \equiv 7\ mod\ 11$
2. način: $27 \cdot (55 + 63) \equiv 5 \cdot (0 + 8) \equiv 5 \cdot 8 \equiv 7\ (mod\ 11)$

Izračuna po obeh postopkih sta enaka, saj so števila 27 in 5, 55 in 0, 63 in 8 v istem kongruenčnem razredu. Če sproti med števili izvajamo operacijo *mod m,* kjer *m* predstavlja poljubno pozitivno celo število, lahko prihranimo veliko časa. Pri izbiri elementa iz kongruenčnega razreda po dogovoru izberemo takšen ostanek *r*, ki je $0 \leq r \leq m - 1$. To je posebej uporabno v kriptografiji, kjer imamo pogosto opravka z več sto mestnimi števili.

Če pri izvajanju operacije *mod* nad enačbo vedno izberemo ostanek *r*, ki je $0 \leq r \leq m - 1$, ugotovimo, da smo vedno znotraj množice celih števil, katere elementi so vsa cela števila od 0 do vključno $m - 1$. Takšna množica tvori celoštevilski obroč.

DEFINICJA 3.2 Celoštevilski obroč. Celoštevilski obroč $\mathbb{Z}_m$, kjer $m \in \mathbb{Z}^+$ sestoji iz:

- množice $\mathbb{Z}_m = \{0, 1, \dots, m - 1\}$,

- operacij seštevanja in množenja $\forall a, b \in \mathbb{Z}_m$, tako da:
  - $a + b \equiv c \bmod m\,, (c \in \mathbb{Z}_m)$
  - $a \cdot b \equiv d \bmod m\,, (d \in \mathbb{Z}_m)$

Po definiciji 3.2 sta v $\mathbb{Z}_m$ definirani le dve računski operaciji. Tako deljenje za to množico ni definirano. Znak za deljenje v množici $\mathbb{Z}_m$ predstavlja multiplikativni inverz.

DEFINICIJA 3.3 Multiplikativni inverz. Naj bo $a \in \mathbb{Z}_m$, multiplikativni inverz a^{-1} je definiran kot

$$a \cdot a^{-1} \equiv 1 \bmod m.$$

Multiplikativni inverz ni definiran za vse elemente, vendar obstaja samo za tiste elemente, kjer je *D(a, m) = 1*. Evklidov algoritem, ki se uporablja za učinkovito iskanje inverzov, je opisan v naslednjem poglavju. [6]

3.2 Razširjen Evklidov algoritem

Evklidov algoritem je najučinkovitejši znan postopek, s katerim lahko poiščemo največji skupni delitelj dveh števil. Definicija Evklidovega algoritma je povzeta po [7].

DEFINICJA 3.4 Evklidov algoritem. Dani sta dve od nič različni pozitivni celi števili a in b. Predpostavimo tudi, da je $a \geq b$ in določimo $a = r_1$ in $b = r_2$. Potem z deljenjem izračunamo taka cela števila $q_1, q_2, \ldots$ in $\mathrm{r}_1, r_2, \ldots$, da velja

$$\begin{aligned} n &= 3 \qquad r_1 = q_1 r_2 + r_3 \\ n &= 4 \qquad r_2 = q_2 r_3 + r_4 \\ n &= 5 \qquad r_3 = q_3 r_4 + r_5 \\ &\vdots \end{aligned}$$

in $\boldsymbol{r_1 > r_2 > \cdots \geq 0}$**.** V splošnem lahko Evklidov algoritem še posplošimo in ga zapišemo tudi kot aritmetično zaporedje

$$\{r_{n-2} = q_{n-2} r_{n-1} + r_n\}_{n \in \mathbb{N};\, n=3}$$

Algoritem se ustavi, ko je $r_n = 0$. Največji skupni delitelj dobimo tako, da preberemo vrednost r_{n-1}, ko je $r_n = 0$. Ker je naravnih števil manjših od b končno mnogo, se bo algoritem zagotovo ustavil v končno mnogo korakih.

Iz Evklidovega algoritma lahko izpeljemo tudi razširjeno verzijo, s katerim poleg največjega skupnega delitelja števil *a, b* izračunamo celi števili *x, y*, ki zadostita Bezoutovi identiteti. V nadaljevanju je predstavljena definicija o razširjenem Evklidovem algoritmu. Definiciji sta povzeti po [8].

DEFINICIJA 3.5 Bezoutova identiteta. Naj bosta $a, b \in \mathbb{Z}^+$, ki imata največji skupni delitelj $d \in \mathbb{Z}^+$. Potem $\exists x, \exists y \in \mathbb{Z}$, ki zadovoljita enačbo $ax + by = d$. [8]

DEFINICIJA 3.6 Razširjen Evklidov algoritem. Predpostavimo, da je $a > b$ in $a, b \in \mathbb{Z}^+$.

Definirajmo še zaporedje $\{q_i\}, \{r_i\}, \{s_i\}, \{t_i\}$, kjer $r_o = a$ in $r_1 = b$. Vrednost $i \leftarrow 2$ in *i* nato povečujemo ob koncu vsake iteracije. V vsaki iteraciji bomo našli q_i, r_i, s_i, t_i tako da $r_{i-2} = q_i r_{i-1} + r_i$, $0 < r_i < r_{i-1}$. Prav tako pa si želimo zapisati r_i kot linearno kombinacijo števil *a* in *b*, kjer $r_i = s_i\, a + t_i b$. Ampak $r_i = \ r_{i-2} - q_i r_{i-1}$, torej preoblikujemo enačbo v

$$r_i = s_{i-2}a + t_{i-2}b - (s_{i-1}a + t_{i-1}b)q_i = (s_{i-2} - s_{i-1}q_i)a + (t_{i-2} - t_{i-1}q_i)b.$$

Iz te enačbe lahko razberemo $s_i = s_{i-2} - s_{i-1}q_i$ in $t_i = \ t_{i-2} - t_{i-1}q_i$.

Potrebno je najti še začetne vrednosti zaporedja $\{s_i\}$ in $\{t_i\}$. Po definiciji r_i dobimo

$$a = \ r_0 = s_0 a + t_0 b \Rightarrow s_0 = 1, t_0 = 0$$

$$b = \ r_1 = s_1 a + t_1 b \Rightarrow s_1 = 0, t_1 = 1.$$

Algoritem izvajamo dokler vrednost $r_{i-1} \neq 0$. Algoritem vedno zahteva končno število korakov, saj se zaporedje z vsakim korakom manjša. [9]

Razširjen Evklidov algoritem je še posebej uporaben, ko želimo najti multiplikativni inverz nekega števila. Za lažje razumevanje iskanja inverza najprej preoblikujmo zgornjo enačbo v $ax + my = D(a, m)$. Da v celoštevilskem obroču $\mathbb{Z}_m$ obstaja multiplikativni inverz, mora biti največji skupni delitelj enak ena, torej predpostavimo, da sta si števili *a* in *m* tuji. Če sedaj nad to enačbo izvedemo operacijo *mod m,* dobimo naslednjo enačbo: $ax \equiv D(a, m)\ mod\ m$**.** Ker smo predpostavili, da sta števili *a* in *m* tuji, lahko $D(a, m)$ nadomestimo z 1. Tako dobimo enačbo $ax \equiv 1\ mod\ m$, ki pa je zelo podobna enačbi iz definicije 3.3. V tej enačbi *x* predstavlja a^{-1} oziroma multiplikativni inverz števila *a*.

Primer: poiščimo multiplikativni inverz števila 15 v množici $\mathbb{Z}_{26}$.

Za iskaje inverza začnemo z razširjenim Evklidovim algoritmom. Iščemo vrednost x, ki zadosti enačbo $D(26,15) \equiv 15x\ (mod\ 26)$.

$$26 = 1 \cdot 15 + 11 \qquad 11 = 1 \cdot 26 - 1 \cdot 15$$

$$15 = 1 \cdot 11 + 4 \qquad 4 = 1 \cdot 15 - 1 \cdot 11 = -1 \cdot 26 + 2 \cdot 15$$

$$11 = 2 \cdot 4 + 3 \qquad 3 = 1 \cdot 11 - 2 \cdot 4 = 3 \cdot 26 - 5 \cdot 15$$

$$4 = 1 \cdot 3 + 1 \qquad 1 = 1 \cdot 4 - 1 \cdot 3 = -4 \cdot 26 + 7 \cdot 15$$

Iz zadnje vrstice razberemo enačbo $1 = -4 \cdot 26 + 7 \cdot 15$. Sedaj nad enačbo izvedemo *mod 26* in dobimo $1 \equiv 7 \cdot 15\ (mod\ 26)$. Iz tega razberemo, da je multiplikativni inverz števila 15 v $\mathbb{Z}_{26}$ enak 7.

3.3 Grupe

Grupa je ena od osnovnih sestavnih delov abstraktne algebre, prav tako pa na njej temeljijo skoraj vsi asimetrični kriptografski sistemi. V primeru kriptografije z eliptičnimi krivuljami uporabljamo grupe, ko imamo opravka z eliptičnimi krivuljami nad končnimi polji. Za lažje razumevanje pa si najprej oglejmo definicijo grupe.

DEFINICIJA 3.7 Grupa. Grupa G je par (G, ◦), kjer je G neprazna množica in ◦ asociativna dvočlena operacija nad G: $G \circ G \to G$, ki ga imenujemo operacija grupe in vsakemu urejenemu paru $(a, b) \in G$ priredi natanko en element $a \circ b \in G$. Operacija ◦ mora zadoščati aksiomom grupe:

- množica G je zaprta, torej $\forall a, \forall b \in G$, velja $a \circ b \in G$ (zakon o zaprtosti),
- velja asociativnost, torej $\forall a, \forall b$ in $\forall c \in G$, velja $(a \circ b) \circ c = a \circ (b \circ c)$ (zakon o asociativnosti),
- v G obstaja takšen nevtralni element e, za katerega $\forall a \in G$ velja $e \circ a = a \circ e = a$ (zakon o identiteti),
- $\forall\ a \in G$ obstaja inverzni element $a^{-1} \in G$, za katerega velja $a \circ a^{-1} = a^{-1} \circ a = e$ (zakon o inverzu),
- velja komutativnost, torej $\forall a, \forall b \in G$, velja $a \circ b = b \circ a$ (zakon o komutativnosti).

Prve štiri lastnosti morajo veljati za vse grupe, med tem pa je peta lastnost pomembna le, če govorimo o Abelovih grupah. [10]

Da lahko grupo uporabimo za kriptografski sistem pa je pomembno tudi, da je grupa končna ter ciklična. V naslednjih dveh definicijah so podrobneje definirane končne in ciklične grupe.

DEFINICJA 3.8 Končna grupa. Grupa (G, ∘) je končna, če ima končno število elementov. Število elementov grupe označimo kot $|G|$. [6]

DEFINICIJA 3.9 Podgrupa. Podgrupo $\langle a \rangle$ imenujemo ciklična podgrupa grupe G generirane z a. Če je $G = \langle a \rangle$ za nek $a \in G$, potem pravimo, da je G ciklična grupa in da je a generator grupe G. Vsaka ciklična grupa ima lahko več generatorjev. Čeprav ima niz $\dots, a^{-2}, a^{-1}, a^0, a^1, a^2, \dots$ neskončno mnogo elementov, ima grupa $\{\mathrm{a}^{\mathrm{n}};\ n \in \mathbb{Z}\}$ končno mnogo elementov preden se le-ti začnejo ponavljati. [11]

DIFENICIJA 3.10 Končne ciklične grupe. Naj bo *G* končna ciklična grupa. Potem velja:

1. število generatorjev grupe *G* označimo $\phi(|G|)$, ta funkcija kot vhodni podatek prejme pozitivno celo število in kot vrednost funkcije vrne število števil, ki so si tuja z vhodom,
2. če je $|G|$ praštevilo, potem so vsi elementi razen števila 1 generatorji. [6]

4 PROBLEM DISKRETNEGA LOGARITMA

Problem diskretnega logaritma predstavlja osnovo, na kateri so zgrajeni ostali kriptografski algoritmi, med drugimi so to RSA, izmenjava ključev Diffie-Hellman, ElGamal in nenazadnje tudi eliptične krivulje. V abstraktni algebri diskretni logaritem predstavlja analogijo za navadne logaritme znotraj grup. Navadni logaritem je definiran kot $\log_a b = x$, kjer sta $a, b \in \mathbb{R}^+$ in $a \neq 1$, njegova rešitev pa reši enačbo $a^x = b$. Skozi čas smo razvili učinkovite načine, kako lahko rešujemo logaritemske enačbe znotraj množice realnih in kompleksnih števil, vendar (še) ne poznamo učinkovitega algoritma, ki reši logaritem znotraj končne ciklične grupe. V kriptografiji je torej uporaben, ker zlonamerni osebi oteži iskanje rešitve enačbe, torej je lahko *b* uporabljen kot javni ključ, izračun tega elementa pa je preprost, če poznamo *a* in *x.* V tem primeru število x tako predstavlja zasebni ključ. [12]

4.1 Izmenjava ključev Diffie-Hellman

Algoritem za izmenjavo ključev Diffie-Hellman je predstavljal revolucijo na področju kriptografije. Z javno objavo leta 1976 je omogočil, da sta se lahko dve osebi, pogosteje dva računalnika, dogovorila o skupnem ključu, ki ga bosta uporabila za nadaljnjo šifriranje in dešifriranje z enim od že obstoječih simetričnih kriptografskih algoritmov, to so tisti algoritmi, ki zahtevajo uporabo enakega ključa za enkripcijo kot tudi dekripcijo. Velika prednost tega algoritma je tudi v tem, da je programska implementacija relativno preprosta, njegovo delovanje ne zahteva veliko procesorske moči, prav tako pa začetni parametri sprva sploh niso nujno poznani obema računalnikoma, saj javna objava le-teh ne vpliva na stopnjo varnosti. V poglavju 2.1 je bila izmenjava ključev predstavljena na preprostem primeru, v nadaljevanju pa bo predstavljen dejanski algoritem za izmenjavo ključev Diffie-Hellman. Definicija 4.1 je povzeta po [13].

DEFINICIJA 4.1 Algoritem za izmenjavo ključev Diffie-Hellman. Ana in Bine se želita dogovoriti za skupen ključ, ki ga bosta uporabljala kot šifrirni ključ za nadaljnjo komunikacijo. Postopek:

1. Ana in Bine se dogovorita za skupna parametra, torej izbereta končno ciklično grupo *G*, kjer je binarna operacija grupe predstavljena multiplikativno, in enega od generatorjev te grupe *g*, kjer je $g \in G$. Ta parametra sta lahko javno znana.

2. Ana nato izbere poljubno naravno število *a,* ki predstavlja njen zasebni ključ, ki mora ostati skrivnost*,* nato izračuna vrednost g^a ter jo pošlje Binetu.

3. Bine prav tako izbere poljubno naravno število *b,* ki predstavlja njegov zasebni ključ, nato izračuna vrednost g^b ter jo pošlje Ani.
4. Ana prejeme število g^b in izračuna vrednost $(g^b)^a$.
5. Bine prav tako prejme število g^a in izračuna vrednost $(g^a)^b$.

Ker aksiomi grupe zahtevajo, da je vsaka grupa asociativna, lahko iz tega sklepamo, da sta vrednosti $(g^b)^a$ in $(g^a)^b$ enaki. S tem postopkom sta se dogovorila za enaki številski vrednosti, ki ju lahko uporabita kot ključ za šifriranje in dešifriranje simetričnega kriptografskega algoritma.

Za večjo varnost je dobro, da ima uporabljena končna ciklična grupa veliko število elementov, sploh ker imamo opravka z grupo $\mathbb{Z}p^*$, proti kateri obstaja dokaj učinkovit algoritem za napad imenovan GNSF (General Number Field Sieve).

Primer: Ana in Bine se dogovorite za javne parametre, in sicer končno ciklično grupo $\mathbb{Z}_{11}^* = \{1,\ 2, \dots, 10\}$, ki je bila generirana iz $\mathbb{Z}_{11}$ z generatorjem *g* = 2.

Ana	**Bine**
a = 4	b = 17
$A = g^a = 2^4 \equiv 5\ (mod\ 11)$	$B = g^b = 2^{17} \equiv 7\ (mod\ 11)$
$K = B^a = 7^4 \equiv 3\ (mod\ 11)$	$K = A^b = 5^{17} \equiv 3\ (mod\ 11)$

Iz zgornjega primera lahko vidimo, da velja enakost $(g^b)^a = (g^a)^b$. Prav tako pa je izbira zasebnega ključa posameznika povsem poljubna, saj se nahajamo znotraj ciklične grupe, kjer na primer za $\mathbb{Z}_{11}^*$ velja, da je $2^2 \equiv 2^{12} \equiv 2^{22} (\mathrm{mod}\, 11)$.

Prav tako se poraja vprašanje, zakaj Eva, ki želi prav tako vedeti njun skupni ključ in s tem prisluškovati njunim pogovorom, ne more iz javnih parametrov, torej $\mathbb{Z}_{11}^*$, generatorja *g*, ter A in B, razbrati Aninega in Binetovega zasebnega ključa. Če bi Eva lahko izračunala zasebni ključ le enega od njiju, njun skupni ključ ne bi bil več varen. Zapišimo enačbo z vsem kar Eva pozna, torej $5 = 2^a$ in $7 = 2^b$. Ker iščemo vrednost eksponente, bi se te enačbe verjetno lotila s pomočjo logaritma, vendar ne smemo pozabiti, da se nahajamo znotraj končne ciklične

grupe. Torej če bi želeli rešiti to enačbo s pomočjo logaritmov, bi morali rešiti problem diskretnega logaritma, kar pa predstavlja nepremagljivo oviro tudi za najzmogljivejše superračunalnike, saj trenutno ne poznamo učinkovitega algoritma za reševanje diskretnega logaritma. Ena izmed preprostejših možnosti za rešitev diskretnega logaritma nam predstavlja uporabo napada s surovo silo (*ang. brute force attack*), kjer na mesto *a* in *b* vstavljamo vsa možna naravna števila, dokler ne pridemo do pravilnega odgovora. V zgornjem primeru bi do rešitve prišli zelo hitro, vendar če bi imeli opravka z 2048-bitnimi števili, bi bilo rešitev veliko težje izračunati. Matematiki so razvili tudi dva učinkovitejša algoritma za reševanje problema diskretnega logaritma, in sicer ''mali korak, veliki korak'' ter Pollardov RO algoritem za logaritme. Ta algoritma sta sicer učinkovitejša kot napad s surovo silo, vendar sta še vedno prepočasna pri daljših ključih.

4.2 Posplošen diskretni logaritem

V predhodnem poglavju smo ugotovili, da varnost algoritma za izmenjavo ključev Diffie-Hellman temelji na nezmožnosti napadalca, da reši diskretni logaritem in tako pridobi Anin ali Binetov zasebni ključ. Če bi lahko problem diskretnega logaritma posplošili, bi ga lahko uporabili tudi v ostalih kriptografskih algoritmih in tako ne bi bili omejeni le na takšne, ki delujejo znotraj končne multiplikativne ciklične grupe $\mathbb{Z}_{11}^*$. Izkaže se, da se problem diskretnega logaritma pojavlja znotraj vsake končne ciklične grupe, vendar so nekateri veliko lažje rešljivi kot drugi.

DEFINICIJA 4.2 Posplošen problem diskretnega logaritma. Dano imamo neko končno ciklično grupo *G* in elementa *g* in *h*, kjer je $g \in G$ in $h \in \langle g \rangle$. Najdi takšen $x \in \mathbb{Z}^+$, ki reši enačbo $h = g^x$.

Na podlagi definicije 4.2 ločimo štiri vrste posplošenega diskretnega logaritma, ki se uporabljajo v kriptografiji:

- Multiplikativne grupe $\mathbb{Z}_p$, kjer je *p* praštevilo in njene ciklične podgrupe $\mathbb{Z}_p^*$. Poleg algoritma za izmenjavo ključev Diffie-Hellman to vrsto diskretnega logaritma uporablja tudi šifriranje Elgamal in DSA (Digital Signature Algorithm).
- Multiplikativne grupe Galoisovih polj $GF(2^m)$ in podgrupe, ki jih tvori. Problem diskretnega logaritma je zelo podoben, kot tisti v prejšnji točki, vendar obstajajo učinkovitejši algoritmi za napad na kriptografske sisteme z Galoisovimi polji, zato so v

praksi redko uporabljeni. Če bi Galoisova polja želeli uporabljati kot varni kriptografski algoritem, bi morali biti ključi še daljši od 2048-bitov, s čimer uporaba postane nepraktična. Kljub slabostim, ki jih ima v asimetrični kriptografiji, pa se uporablja za algoritem AES, kjer s pomočjo ostalih kriptografskih tehnik tvori zelo močen kriptografski sistem.

- Ciklične grupe, ki jih tvorijo eliptične krivulje nad končnimi polji. Uporaba eliptičnih krivulj za kriptografijo narašča iz leta v leto, ker je problem diskretnega logaritma veliko varnejši in odpornejši pred napadi kot tisti v multiplikativnih grupah $\mathbb{Z}_p^*$. To omogoča do 30-krat krajše ključe, ki pa zagotavljajo enako stopnjo varnosti, kot npr. RSA, kjer varnost šifriranja sloni na težavnosti faktorizacije sestavljenih števil na prafaktorje.
- Hipereliptične krivulje, ki imajo podoben problem diskretnega logaritma kot eliptične krivulje in jih lahko dojemamo kot posplošene eliptične krivulje. Trenutno se uporabljajo izključno v akademskih vodah, saj še niso pridobile dovolj prepoznavnosti za širšo rabo. Zaradi trenutno majhnega interesa v raziskavo hipereliptičnih krivulj primanjkuje zanesljivih informacij glede stopnje varnost kriptografije s hipereliptičnimi krivuljami. [6]

5 ELIPTIČNE KRIVULJE

Odkritje eliptičnih krivulj, čeprav implicitno, sega že v antiko, kjer se je s tem problemom ukvarjal Diofant. Diofantov problem je bil kasneje pomemben tudi Fermatu, Eulerju in ostalim matematikom, ki so se ukvarjali s teorijo števil. Sprva na tem področju ni bilo pomembnejših odkritji, zato so se z eliptičnimi krivuljami ukvarjali le manj poznani matematiki. Z dokazom zakona o vzajemnem učinku, ki pravi, da če prvo telo deluje na drugo telo z neko silo, potem tudi drugo telo deluje na prvo z nasprotno enako silo, so eliptične krivulje dobile nov zagon v svetu matematike. S pomočjo eliptičnih krivulj so tako med drugim dokazali Weilovo in Riemannovo domnevo.

Uporaba v kriptografiji se je pričela zahvaljujoč Nealu Koblitzu in Victorju Millerju, ki sta leta 1985 neodvisno predlagala, da bi lahko pri izračunih diskretnega logaritma uporabili grupo na eliptičnih krivuljah nad končnim obsegom. Glavna prednost kriptografije z eliptičnimi krivuljami je dolžina ključa, ki je približno trikrat krajša, v primerjavi z RSA šifriranjem, katere varnost temelji na domnevi, da ne obstaja učinkovit algoritem za faktorizacijo velikih števil na prafaktorje. Primerjalna tabla dolžin ključev je predstavljena v nadaljevanju.

Tabela 1: Prikaz dolžine ključev pri različnih kriptografskih algoritmih [14]

Simetrične šifre (AES)	Asimetrične šifre (RSA)	Eliptične krivulje
40 bitov	274 bitov	80 bitov
80 bitov	1024 bitov	160 bitov
192 bitov	8192 bitov	409 bitov
256 bitov	16384 bitov	540 bitov

5.1 Eliptične krivulje nad $\mathbb{R}$

DEFINICIJA 5.1 Splošna oblike enačbe za eliptične krivulje. Eliptično krivuljo $E(F)$ definiramo kot množico točk v polju *F*, ki zadovolji naslednjo enačbo:

$$y^2 + a_1xy + a_2y = x^3 + a_3x^2 + a_4x + a_5,$$

kjer so $a_1, a_2, \dots, a_5 \epsilon F$. Karakteristika polja mora biti različna od 2 in 3, saj ne smemo dopustiti možnosti več ničel. Predpostaviti moramo torej, da

$4a^3 + 27b^2 \neq 0$. Tako dobimo preprostejšo obliko zapisa eliptične krivulje, imenovano tudi Weierstrassova normalna oblika:

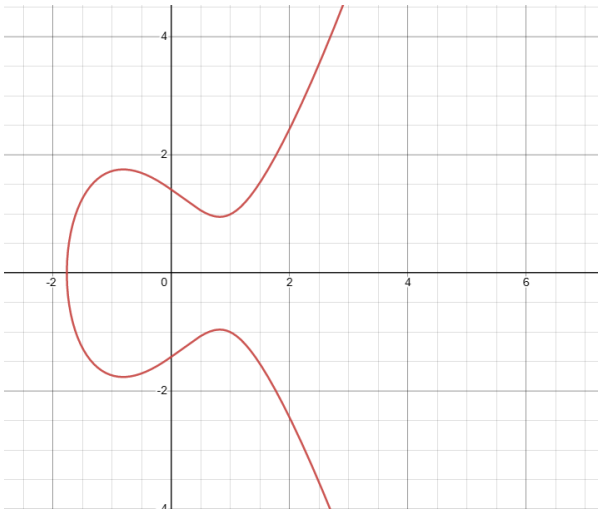
$$y^2 = x^3 + ax + b,$$

kjer sta $a, b \in \mathbb{R}$. [15]

Eliptično krivuljo *E* sestavljajo vse točke (x, y), ki ustrezajo Weierstrassovi enačbi in posebna točka neskončnosti (∞, ∞), ki jo lahko označimo tudi z 0. Njen obstoj je pomemben, saj predstavlja nevtralni element, ki v grupi mora obstajati, da zadovolji njene aksiome. Ker je ta točka zelo abstraktna, si jo lahko predstavljamo kot točko, ki hkrati leži pri pozitivni neskončnosti in hkrati negativni neskončnosti oziroma drugače povedano, ta točka povezuje pozitivno ter negativno neskončnost. V splošnem torej množico vseh točk zapišemo:

$$E(F) = \{(\infty, \infty)\} \cup \{(x, y) \in F^2;\ y^2 = x^3 + ax + b\}.$$

Ker si eliptično krivuljo težko predstavljamo, je na spodnji sliki prikazan primer eliptične krivulje $y^2 = x^3 + 7$ nad $\mathbb{R}^2$. Ker je y kvadriran, to pomeni, da tako vedno kot rešitev dobimo pozitivno in negativno vrednost y, torej je krivulja simetrična glede na abscisno os. [16]



Slika 3: Prikaz eliptične krivulje nad realnimi števili

5.2 Zakoni grupe

V poglavju 3.3 smo predstavili algebraično strukturo grup. Med drugim mora v grupi obstajati vsaj ena binarna operacija, ki zadovolji aksiomom grupe. V primeru eliptičnih krivulj prav tako obstaja takšna operacija, vendar se namesto znaka $\circ$ za binarno operacijo pogosteje uporablja znak za seštevanje +. Kljub uporabi znaka + pa je potrebno upoštevati, da je izbira

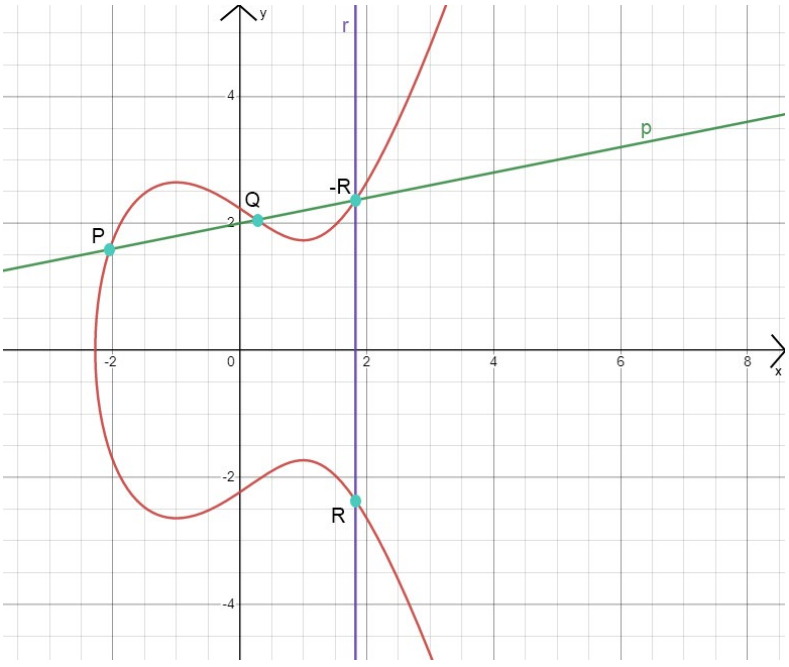
operacije povsem poljubna in nima povezave z operacijo seštevanja kot jo poznamo v množici $\mathbb{R}$.

Za lažjo in učinkovitejšo predstavo binarnih operaciji nad eliptičnimi krivuljami bomo najprej predpostavili, da je krivulja *E* definiran nad poljem $\mathbb{R}$. V množici $E(\mathbb{R})$ lahko iz dveh točk vedno izračunamo tretjo točko. Ta lastnost igra pomembno vlogo pri kriptografiji.

Predpostavimo, da imamo dani dve točki $P(x_1, y_1)$ in $Q(x_2, y_2)$, ki sta obe del $E(\mathbb{R})$. Izračunati želimo tretjo točko *R,* ki je prav tako del $E(\mathbb{R})$ in ustreza naslednji enačbi:

$$P + Q = R \text{ ali } (x_1, y_1) + (x_2, y_2) = (x_3, y_3).$$

Če $x_1 \neq x_2$ – seštevanje točk: Narišimo premico *p*, ki poteka skozi točki *P* in *Q*. Zaradi lastnosti eliptičnih krivulj bo ta premica vedno sekala eliptično krivuljo še v natanko eni točki. Ta točka predstavlja točko *-R.* Če želimo najti *R,* moramo *-R* zrcaliti še čez abscisno os. To storimo tako, da narišemo premico *r*, ki poteka skozi točko *-R* in je hkrati vzporedna z ordinatno osjo ter poiščemo presečišče z eliptično krivuljo. Za lažje razumevanje je postopek na spodnji sliki predstavljeno še grafično.



Slika 4: Seštevanje dveh različnih točk

Oglejmo si še matematični postopek računanja. Najprej je potrebno zapisati enačbo premice, ki poteka skozi točki *P* in *Q,* torej izračunamo smerni koeficient $k = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$. Nato je

potrebno izračunati še vrednosti x in y, ki sta koordinati točke *R*. Najprej izpeljimo enačbo za izračun abscise tretje točke.

Če želimo najti koordinate tretje točke, moramo najti presečišče premice s predpisom $y = kx + n$ in krivulje s predpisom $y^2 = x^3 + ax + b$, zato ju enačimo med seboj, torej:

$$y^2 = y^2$$

$$(kx + n)^2 = x^3 + ax + b$$

$$x^3 - k^2x^2 + (a - 2kn)x + (b + n^2) = 0$$

Tako dobimo enačbo polinoma tretje stopnje, ki pa mora imeti tri rešitve, če želimo najti tretje presečišče, zato zapišemo kubično enačbo v ničelni obliki, torej:

$$a(x - x_1)(x - x_2)(x - x_3) = 0 \ /: a, \text{ kjer } a \neq 0$$

$$(x^2 - (x_2 + x_1)x + x_1x_2)(x - x_3) = 0$$

$$x^3 - x^2x_3 + (-x_1 - x_2)x^2 + (x_1x_3 + x_2x_3)x + x_1x_2x - x_1x_2x_3 = 0$$

$$x^3 - (x_1 + x_2 + x_3)x^2 + (x_1x_2 + x_1x_3 + x_2x_3)x - x_1x_2x_3 = 0$$

S tem smo dobili dve kubični enačbi, ki ju lahko enačimo med seboj. Če se osredotočimo na koeficient prve in druge enačbe pri x^2, opazimo, da lahko iz njiju izpeljemo enačbo za x_3, torej zapišemo:

$$k^2 = x_1 + x_2 + x_3$$

$$x_3 = k^2 - (x_1 + x_2)$$

S tem smo izpeljali formulo za izračun abscise. Izpeljimo še formulo za izračun ordinate nove točke. Formulo v tem primeru izpeljemo iz formule za izračun strmine premice, ki poteka skozi točki P in R.

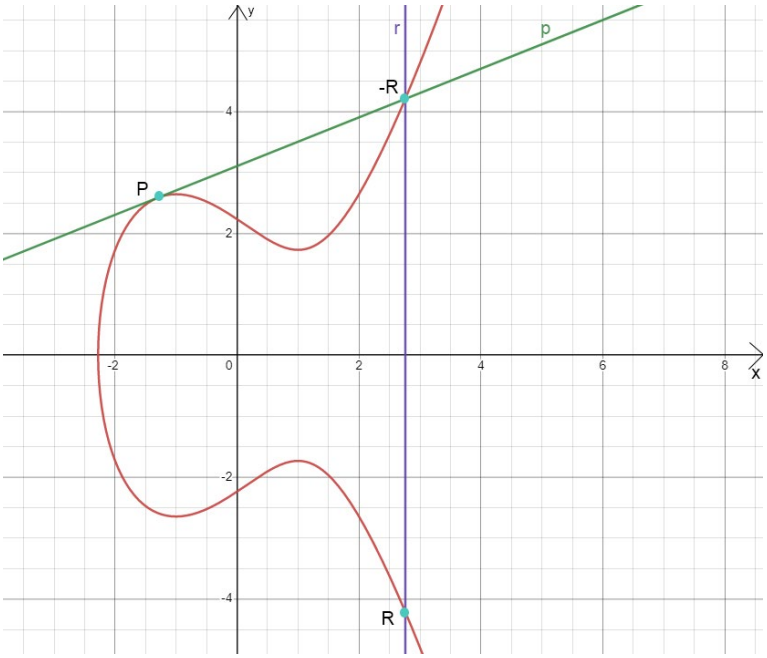
$$k = \frac{y_3 - y_1}{x_3 - x_1}$$

$$-y_3 = k(x_1 - x_3) - y_1$$

S tem dobimo enačbo, pri čemer je y_3 negativno predznačen. To je pomembno, saj je potrebno dobljeno ordinato zrcaliti še čez abscisno os, torej je enačba za izračun ordinate nove točke:

$$y_3 = k(x_1 - x_3) - y_1$$

Če $x_1 = x_2$ in $y_1 = y_2$ – podvajanje točke: Če sta *x* koordinati v obeh primerih enaki, lahko iz tega razberemo, da je $P = Q$. Ker imamo dve enaki točki, skozi njiju ne moremo narisati premice, kot smo to lahko storili v prejšnjem primeru. Temu problemu se izognemo z uporabo odvoda in tako izračunamo tangento *p* na krivuljo v dani točki. Ta tangenta bo zaradi lastnosti eliptičnih krivulj prav tako sekala krivuljo v še eni točki, imenovani *-R*. Da dobimo točko *R, -R* zrcalimo čez abscisno os oziroma s pomočjo premice *r*. Zaradi lažjega razumevanja je na spodnji sliki postopek prikazan še grafično.



Slika 5: Podvajaje točke

Oglejmo si še matematični postopek računanja koordinat točke R. Torej prvotno enačbo krivulje $y^2 = x^3 + ax + b$ preoblikujemo v $y = (x^3 + ax + b)^{\frac{1}{2}}$. Sedaj ko smo izrazili *y*, lahko zapišemo enačbo odvoda in ga razrešimo, saj bo enak enačbi za računanje vrednosti smernega koeficienta linearne funkcije, ki potega skozi dano točko, torej:

$$k = \frac{d}{dx}(x^3 + ax + b)^{\frac{1}{2}}$$

$$k = \frac{1}{2}(x^3 + ax + b)^{-\frac{1}{2}} \cdot (3x^2 + a)$$

$$k = \frac{3x^2 + a}{2\sqrt{x^3 + ax + b}}$$

$$k = \frac{3x^2 + a}{2y}$$

Ko smo izračunali *k*, je potrebno izračunati še x_3 in y_3 koordinati. To izračunamo po naslednjih formulah:

$$x_3 = k^2 - 2x_1$$

$$y_3 = k(x_1 - x_3) - y_1.$$

Postopek izpeljave formul za izračun koordinat nove točke je povsem enak izpeljavi pri seštevanju točk. Edina razlika, ki se pojavi pri podvajanju točke je ta, da je $x_1 = x_2$, zato se pri izračunu abscise v formuli pojavi $-2x_1$ in ne $-(x_1 + x_2)$, kot je bilo zapisano pri izpeljavi.

S pomočjo grupnih zakonov pa za lažji zapis definirajmo še skalarno množenje.

DEFINICIJA 5.2 Skalarno množenje. Množenje s skalarjem je prištevanje točk *n*-krat, kjer je $n \in \mathbb{Z}^+$ in *P* točka, ki leži na krivulji.

$$nP = P + P + \cdots + P, \text{ n-kart}$$

5.3 Eliptične krivulje nad končnim poljem $\mathbb{Z}_p$

V predhodnem poglavju smo definirali grupo, ki jo tvori eliptična krivulja, njeno binarno operacijo, prav tako pa smo izpeljali formule za računanje s to binarno operacijo. Zaradi lažje predstave je bilo vse predstavljeno v ravnini $\mathbb{R}^2$, kar pa ni najboljša izbira za kriptografski algoritem. V kriptografiji imamo najpogosteje opravka z operacijami nad končnimi polji, saj so te vrste kriptografskih sistemov najbolje raziskane, njihovo varnost pa zagotavlja (ne)zmožnost rešitve problema diskretnega logaritma. V primeru kriptografije z eliptičnimi krivuljami dosežemo končno polje z operacijo $mod\ p$, kjer *p* predstavlja ogromno praštevilo. Tako imamo v splošnem opravka z eliptično krivuljo, katere enačba je $y^2 \equiv x^3 + ax + b \,(\mathrm{mod}\, p)$, $a, b, p \in \mathbb{R}$. Ker je ta eliptična krivulja definirana nad končnim poljem, je potrebno tudi pri enačbah za računaje smernega koeficienta, koordinate *x* in koordinate *y,* dodati operacijo $mod\ p$. Postopek računanja koordinate tretje točke je tako nekoliko drugačno, zato je postopek računanja predstavljen s spodnjim primerom.

Primer: dana je eliptična krivulja $E: y^2 \equiv x^3 + 2x + 2 \,(\mathrm{mod}\, 17)$ s točkama $P(5,1)$ in $5P(9,16)$. Izračunaj koordinati točke $6P$, ki jo dobimo, če seštejemo točki $P + 5P$.

Preden se lahko lotimo računanja, se moramo prepričati, ali gre za dve enaki ali dve različni točki. To preverimo tako, da primerjamo vrednosti ordinate in abscise obeh točk. Ker se vrednosti razlikujeta, tako ugotovimo, da gre za dve različni točki, zato bomo pri računanju tretje točke uporabljali formule za seštevanje točk.

Najprej se lotimo računanja smernega koeficienta premice, ki poteka skozi dani točki. Uporabimo formulo: $k = \frac{y_2 - y_1}{x_2 - x_1} \equiv \frac{16-1}{9-5} \equiv 15 \cdot 4^{-1} \equiv 15 \cdot 13 \equiv 8\ (mod\ 17)$.

Pri računanju smernega koeficienta je potrebno upoštevati, da je operacija mod definirana v množici $\mathbb{Z}_{17}$, kjer deljenje, kot ga poznamo v množici $\mathbb{R}$, ni definirano. V tem primeru ulomek predstavlja zmnožek vrednosti števca in multiplikativnega inverza imenovalca, torej $\frac{15}{4} \equiv 15 \cdot 4^{-1}\ (mod\ \ 17)$. Da bi našli vrednost multiplikativnega inverza, uporabimo razširjen Evklidov algoritem, torej:

$$17 = 4 \cdot 4 + 1 \qquad 1 = 1 \cdot 17 - 4 \cdot 4$$

$$1 \equiv -4 \cdot 4 \equiv 13 \cdot 4\ (mod\ 17)$$

Iz razširjenega Evklidovega algoritma lahko tako razberemo, da je multiplikativni inverz števila -4 v množici $\mathbb{Z}_{17}$ enak 13.

Ko smo izračunali vrednost smernega koeficienta, moramo v spodnji enačbi vstaviti le še zahtevane podatke in tako izračunamo koordinate točke $6P$.

$$x_3 = k^2 - (x_1 + x_2) \equiv 8^2 - (5 + 9) \equiv 16(mod\ 17)$$

$$y_3 = k(x_1 - x_3) - y_1 \equiv 8(5 - 16) - 1 \equiv 13\ (mod\ 17)$$

Iz zgornjih enačb lahko razberemo, da ima točka $6P$ koordinate $(16, 13)$.

Prav tako se izkaže, da točke na eliptični krivulji nad končnim poljem skupaj s točko neskončnosti tvorijo končno ciklično grupo. Generator te ciklične grupe je vsaka točka na krivulji, saj je izpolnjen drugi pogoj definicije 3.10.

Primer: dano imamo eliptično krivuljo E: $y^2 \equiv x^3 + 2x + 2\ (\mathrm{mod}\ 17)$. Izračunati želimo vse točke, ki so del ciklične grupe z generatorjem $P(5, 1)$.

Iz definicije 3.9 vemo, da najdemo vse elemente ciklične grupe tako, da generator *a* potenciramo na vsa naravna števila *n*, dokler ne generiramo vseh elementov te grupe

$\{a^n;\ n \in \mathbb{Z}\}$. Ker se v primeru eliptičnih krivulj binarna operacija pogosteje predstavlja aditivno kot multiplikativno, bomo generirali vse elemente tako, da bomo generator *a* množili z naravnimi števili *n*, dokler ne generiramo vseh elementov grupe $\{na; n \in \mathbb{N}\}$. V našem konkretnem primeru tako generiramo naslednje točke:

$$1P = (5,1) \qquad 11P = (13,10)$$

$$2P = P + P = (6,3) \qquad 12P = (0,11)$$

$$3P = 2P + P = (10,6) \qquad 13P = (16,4)$$

$$4P = (3,1) \qquad 14P = (9,1)$$

$$5P = (9,16) \qquad 15P = (3,16)$$

$$6P = (16,13) \qquad 16P = (10,11)$$

$$7P = (0,6) \qquad 17P = (6,14)$$

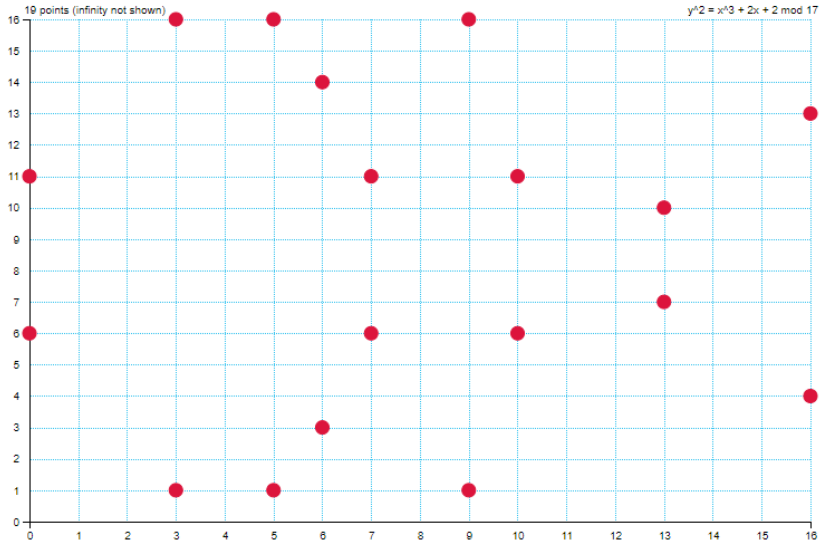
$$8P = (13,7) \qquad 18P = (5,16)$$

$$9P = (7,6) \qquad 19P = (\infty,\infty)$$

$$10P = (7,11)$$

S tem postopkom smo generirali 19 različnih točk, preden bi se te začele ponavljati, torej je $|E| = 19$. Prav tako lahko cikličnost te grupe potrdimo, saj je $20P = 19P + P = (\infty,\infty) + (5,1) = (5,1) = 1P$. Prepričajmo se še, da generirana ciklična grupa sledi vsem aksiomom grupe. Zakon o zaprtosti lahko potrdimo, saj vse točke ležijo na krivulji. Prav tako obstaja nevtralni element, ki je v tem primeru točka $19P$. Za vsako generirano točko obstaja tudi njen multiplikativni (aditivni) inverz, saj $P + 18P = 2P + 17P = 3P + 16P = \cdots = 19P$. Asociativnost in komutativnost operacije zagotavljajo aksiomi grupe.

S podrobno analizo generiranih točk lahko pridemo še do enega pomembnega spoznanja, in sicer te točke si ne sledijo v nekem logičnem zaporedju. Lahko bi rekli tudi, da so te točke razporejene povsem naključno. Na Sliki 6 je prikazan graf s točkami iz ciklične grupe, ki jih tvori eliptična krivulja $E: y^2 \equiv x^3 + 2x + 2 \pmod{17}$, in generatorjem ciklične podgrupe, ki je lahko poljubna točka krivulje *E*. Točka neskončnosti na grafu ni prikazana.



Slika 6: Graf eliptične krivulje E nad končnim poljem $\mathbb{Z}_{17}$

Preden lahko predstavimo kriptografijo z eliptičnimi krivuljami, moramo spoznati še algoritem, ki bo omogočal učinkovito računanje koordinat poljubne točke na krivulji. V zgornjem primeru smo za izračun točke $nP, n\epsilon\mathbb{N}$, uporabili generator $P \in E$ in ga prišteli samemu sebi *n*-krat. Ta pristop deluje pri krivuljah nad manjšimi končnimi polji, vendar algoritem izgubi učinkovitost in hitrost, če imamo opravka z velikimi polji, saj zahtevnost algoritma narašča linearno, torej če bi imeli opravka z eliptično krivuljo nad končnim poljem velikosti $2^{256} - 1$ in bi vsako sekundo lahko izračunali milijon točk, bi še vedno potrebovali 10^{74} let, da bi izračunali vsako točko na krivulji. Za primerjavo lahko vzamemo starost vesolja, ki je staro le $14 \cdot 10^9$ let. Ker je računanje točk s tem algoritmom ne samo neučinkovito, ampak tudi neizvedljivo, obstaja algoritem podvoji in prištej, ki povprečen čas za računanje točke na krivulji zmanjša na povprečno $1{,}5t$, kjer t predstavlja število bitov, ki so potrebni za binarno reprezentacijo števila *n.* Spodnja definicija je povzeta po (Paar, 2010).

DEFINICIJA 5.3 Algoritem podvoji in prištej. Izračunati želimo točko $T = nP; n\epsilon\mathbb{N}$ in $P, T \in E$. Algoritem:

1. Število *n* pretvorimo v njen ekvivalenten binarni zapis, torej $n = \sum_{i=0}^{t} d_i 2^i$, kjer $d_i \in 0{,}1$ in $d_t = 1$.
2. Nato beremo bite od leve proti desni. Za vsak nov bit točko prištejemo sami sebi (točko podvojimo), če pa je bit enak 1, točki prištejemo še točko P. Za lažje razumevaje je spodaj zapis še v psevdokodi:

```
T = (∞, ∞)
za (i = len(n) - 1; i >= 0; i--):
        T = T + T
        če (d_i == 1):
                T = T + P
vrni (T)
```

Primer: izračunati želimo točko $5P$ na eliptični krivulji $E: y^2 \equiv x^3 + 2x + 2 \pmod{17}$ z generatorjem $P(5,1)$.

Najprej pretvorimo $5P$ v binarno število, torej $(101)_2 P$. Sedaj pričnemo izvajati algoritem, tako da beremo bite od leve proti desni in podvajamo ter prištevamo točke:

$d_2 = 1$, ker je to začetni bit ne izvajamo seštevanja ali podvajanja, torej je trenutna točka enaka P,

$d_1 = 0$, ker je bit enak 0, točko le podvojimo, torej P + P = 2P,

$d_0 = 1$, ker je bit enak 1, najprej točko podvojimo, torej 2P + 2P = 4P, nato pa točki prištejemo še P, torej 4P + P = 5P.

S tem smo izračunali točko 5P v manj korakih, kot bi jih potrebovali, če bi uporabili algoritem linearne zahtevnosti. Razlika med algoritmoma bi postala še očitnejša, če bi uporabili večja števila.

5.4 Kriptografija z eliptičnimi krivuljami

Varnost kriptografije z eliptičnimi krivuljami temelji na nerešljivost problema diskretnega logaritma, zato si najprej oglejmo njegovo definicijo.

DEFINICIJA 5.4 Problem diskretnega logaritma eliptičnih krivulj. Dana je eliptična krivulja *E*. Podani sta tudi točki *P* in *T*, kjer je *P* generator ciklične grupe in *T* ena od točk v njej. Problem diskretnega logaritma eliptičnih krivulj je najti naravno število *n*, kjer je $1 \leq n \leq |E|$, da bo veljalo

$$nP = P + P + \cdots + P = T.$$

Iz zgornje definicije lahko razberemo podobnosti s problemom diskretnega logaritma, ki zagotavlja varnost v izmenjavi ključev Diffie-Hellman, in tako sestavimo algoritem Diffie-Hellman za uporabo na eliptičnih krivuljah. Zaradi podobnosti problema diskretnega logaritma je razlika med algoritmoma le v izbiri skupnih parametrov ter v prikazu operacije, ki je v tem primeru aditivna in ne multiplikativna.

DEFINICIJA 5.5 Izmenjava ključev Diffie-Hellman z eliptičnimi krivuljami. Ana in Bine se želita dogovoriti za skupen ključ, ki ga bosta kasneje uporabili za šifriranje. Postopek:

1. Ana in Bine se najprej dogovorita za skupne parametre, torej izbereta enačbo eliptične krivulje *E,* praštevilo *p* ter generator ciklične grupe $P \in E$. Ti parametri so lahko javno poznani.
2. Ana izbere poljubno število a, ki ustreza pogoju $1 < a < |E|$. To število predstavlja njen zasebni ključ. Nato s pomočjo algoritma podvoji in prištej izračuna točko aP ter jo posreduje Binetu.
3. Bine prav tako izbere poljubno število b, ki ustreza pogoju $1 < b < |E|$. To število predstavlja njegov zasebni ključ. Nato izračuna točko bP in jo pošlje Ani.
4. Ana prejeto točko množi s svojim zasebnim ključem in tako dobi točko baP.
5. Bine prejeto točko množi s svojim zasebnim ključem in prav tako dobi isto točko, in sicer abP.

S tem postopkom sta oba izračunala enake koordinate točke, saj je binarna operacija grupe komutativna, torej velja enakost $a(bP) = b(aP)$. Za šifriranje se običajno uporabi abscisa izračunane točke kot ključ za nadaljnjo šifriranje. Izračunana ordinata se lahko zavrže, saj jo lahko oba s pomočjo abscise izračunata kadarkoli.

V zgornji definiciji je bila izbira enačbe eliptične krivulje predstavljena povsem trivialno, vendar v kriptografiji izven učnih primerov izbira ni tako preprosta, saj vse krivulje ne prinašajo isti nivo varnosti. Prvo merilo, ki ga moramo upoštevati pri izbiri primerne krivulje, je število ničel, ki more biti enako 1. Za izbrano eliptično krivulje se je potrebno prepričati tudi, da ne podleže kakšni od ranljivosti, ki bi napadalcem omogočila lažji napad na našo enkripcijo in tako ogrozila njeno varnost. Izbira primerne krivulje tako predstavlja zahtevno nalogo, tudi za najboljše kriptografe, zato se priporoča uporaba standardiziranih krivulj, katerih varnost je bila temeljito preverjena. Ameriška agencija za kibernetsko varnost NSA je objavila seznam

varnih in preverjenih krivulj, vendar je njihova uporaba odsvetovana, saj obstaja verjetnost, da agencija pozna ranljivost teh krivulj, ki jih ni razkrila javnosti in si tako pridobila orodje za vohunjenje. Ker je kriptografija področje, kjer je pretirana skrb za varnost priporočljivejša kot malomarnost, je priporočljiva uporaba krivulj, ki so bile testirane in preverjene s strani več neodvisnih raziskovalcev in organizacij. Seznam vseh varnih in preizkušenih krivulj je objavljen na spletni strani sefecurves.cr.yp.to.

Z zgornjo definicijo smo definirali kriptografski algoritem, s pomočjo katerega se lahko dva neznanca na varen način dogovorita za skupen kriptografski ključ, ki ga bosta uporabila za enkripcijo pri nadaljnji komunikaciji, vendar morata za enkripcijo uporabiti drug algoritem, najpogosteje AES ali DES. Uporaba simetričnega algoritma za nadaljnjo enkripcijo ima tako prednosti kot slabosti. Prednost je predvsem povečana varnost, saj s ključi enake dolžine dosežemo večjo varnost pri simetričnih algoritmih v primerjavi z asimetričnih algoritmih. S krajšimi ključi dosežemo tudi razbremenitev strojne opreme. Eno od slabosti tega načina pa predstavlja potreba po uporabi dveh različnih kriptografskih algoritmov, zato so raziskovalci razvili kriptografski algoritem ElGamal, kjer imata osebi, ki si želita izmenjati neko sporočilo, dva različna ključa, in sicer javni ter zasebni ključ, in se tako izogneta uporabi dveh različnih kriptografskih algoritmov. V nadaljevanju je predstavljena kriptografija ElGamal.

DEFINICIJA 5.6 Kriptografski algoritem ElGamal z uporabo eliptičnih krivulj. Bine želi Ani varno poslati sporočilo *s*. Za to uporabi naslednji postopek:

1. Ana in Bine se najprej dogovorita za skupne parametre, torej si izbereta enačbo eliptične krivulje *E,* veliko praštevilo *p* ter generator ciklične grupe $P \in E$. Ti parametri so lahko javno poznani.
2. Ana generira svoj zasebni ključ $a \in \mathbb{Z}^+$, ki ga najpogosteje generira s pomočjo generatorja naključnih števil, in s pomočjo zasebnega ključa izračuna svoj javni ključ tako, da izračuna koordinate točke $A = aP$. Generiran javni ključ A bo uporabljen za enkripcijo besedila, ki ga tretja oseba želi varno posredovati Ani, zato Ana Binetu posreduje svoj javni ključ.
3. Bine prejme Anin javni ključ in izbere poljubno naravno število *k*. Da bi originalno sporočilo *s* pretvoril v zašifrirano besedilo *S,* Bine izračuna $K = kP$ in $C = kA + S$ in dobljena rezultata posreduje Ani.

4. Ana s pomočjo prejetih parametrov dešifrira sporočilo z uporabo formule $s = C - aK = C + a(-K)$, kjer *-K* predstavlja nasprotno točko točke *K*, torej moramo ordinato točke *K* negativno predznačiti.

O pravilnosti izvedene operacije se lahko prepričamo s pomočjo $s = C - aK = kA + S - aK = kaP + S - akP$.

6 ODPORNOST ELIPTIČNIH KRIVULJ PRED KVANTNIMI RAČUNALNIKI

V zadnjih letih se je področje kvantnega računalništva pričelo razvijati z neverjetno hitrostjo, čeprav je še vedno v zgodnji fazi razvoja. Zahvaljujoč razvoju na tem področju, lahko sedaj veliko učinkoviteje faktoriziramo sestavljena števila na njihove prafaktorje, simuliramo kvantne sisteme, programiramo vedno večje nevronske mreže in še bi lahko naštevali. Kvantni računalniki bodo v prihodnosti zaradi svoje zmogljivosti predstavljali veliko konkurenco klasičnim računalnikom, kar sprva zveni obetavno, vendar bomo morali zaradi njihove učinkovitosti na novo premisliti marsikaj, med drugim tudi kriptografijo.

Zaradi nevarnosti, ki jo bodo kvantni računalniki predstavljali trenutno uporabljeni kriptografiji, si bomo v šestem poglavju ogledali delovanje kvantih računalnikov in s pridobljenim znanjem kritično ocenili varnost kriptografije z eliptičnimi krivuljami pred napadi kvantnih računalnikov.

6.1 Kvantni računalnik

Mnogim je pojem kvantni računalnik poznana beseda, vendar večina ne razume njihovega delovanja. Glavna razlika med klasičnimi ter kvantnim računalniki je uporaba različne enote za pomnjenje podatkov. Klasični računalniki uporabljajo bite, in sicer en bit predstavlja podatek o opazovanem objektu, ki je lahko 1 ali 0, medtem ko kvantni računalniki uporabljajo kvantne bite, znane tudi kot kubite, kjer en kubit hkrati predstavlja vrednost 1 in 0. Sprva se ta fenomen sliši nesmiselno, vendar je ta v kvantni fiziki poznan kot superpozicija.

Kvantni računalniki sledijo dvema zakonoma kvantno-mehanskega sveta, in sicer superpoziciji ter prepletenosti. Kvantna superpozicija je temeljni princip v kvantni mehaniki in v primeru kvantnih računalnikov predstavlja kvantno logično stanje kubita. To pomeni, da lahko kubiti istočasno obstajajo v več kot enem stanju, čeprav so en sam delec. Zaradi superpozicije kvantni računalniki veliko hitreje upravljajo z veliko večjo količino podatkov, sploh v primerjavi s klasičnimi računalniki. Za delovanje kvantnih računalnikov je pomembno tudi razumevanje kvantne prepletenosti, ki v kvantni mehaniki označuje stanje, ko sta dva delca med seboj odvisna tako, da vrednost enega vpliva na stanje drugega, torej če enemu od povezanih delcev izmerimo stanje 1, bo drugi delec vedno v stanju 0, ne glede na oddaljenost teh dveh delcev. Kvantna prepletenost igra pomembno vlogo v številnih algoritmih za kvantne računalnike,

med drugim pri korekciji napak; ta lastnost je zelo uporabna predvsem pri kvantni kriptografiji. [17]

6.2 Razvoj kvantnih računalnikov

Začetek razvoja kvantnih računalnikov sega v sredino 20. stoletja, ko so fiziki pričeli uporabljati kvantno-mehanski model v novi vrsti računalnikov, s tem so zamenjali bite za kubite. Leta 1980 je Paul Benioff predstavil model kvantno mehanskega Turingovega stroja. V tistem času je mnogo člankov trdilo, da je realizacija kvantnega računalnika nemogoča, a je Benioff v svojem znanstvenem članku to tudi teoretično dokazal. Nekaj let kasneje je teoretični fizik Richard Feynman zatrdil, da ima kvantni računalnik potencial za simuliranje fizičnih pojavov, katerih klasični računalniki niso zmožni. Ta ideja je s podporo različnih člankov drugih fizikov in matematikov pritegnila pozornost ter tako postavila trdno podlago za prihodnje razvijanje kvantnega računalništva. Področje se je v prihodnjih letih začelo hitro razvijati in pa tudi implementirati na različnih področjih npr. industrija, bančništvo, kriptografija ter spletna varnost.

Poleg kvantnih računalnikov so se razvili tudi številni kvantnih algoritmov, s katerimi bi izboljšali oz. optimizirali določene naloge. Leta 1996 je Lov Grover izumil Groverjev algoritem, namenjen iskanju elementa v neurejenem seznamu. Groverjev algoritem je eksponentno hitrejši kot katerikoli algoritem namenjen klasičnim računalnikom. Prav tako je leta 1994 Peter Shor razvil prvi algoritem za iskanje prafaktorjev celega števila. Algoritem je bil razvit z namenom napada na asimetrične kriptografske sisteme, zato je podrobneje opisan v naslednjem poglavju. Enega najpomembnejših kvantnih algoritmov pa sta razvila David Deutsch in Richard Jozsa leta 1992, in sicer Deutsch-Jozsa algoritem, ki je prvi dokazal, da so nekateri problemi veliko težji za klasične računalnike kot kvantne računalnike ter obratno. [18]

6.3 Shorov algoritem

Ameriški matematik Peter Shor je leta 1994 razvil prvi poznani kvantni algoritem za iskanje prafaktorjev sestavljenega števila. Algoritem deluje na principu iskanja dveh prafaktorjev poljubnega števila *N,* vendar je pri uporabi Shorovega algoritma potrebno upoštevati, da z njim ne moremo faktorizirati vseh števil, vendar le števila, ki ustrezajo vsem naslednjim kriterijem:

- število je sestavljeno število,
- število je liho,
- število ne sme biti mogoče zapisati kot potenco.

Postopek algoritma:

1. Izberemo poljubno število *N*, ki ga želimo faktorizirati, in ustreza zgoraj navedenim pogojem.
2. Izberimo poljubno število *k,* kjer je $1 < k < N$.
3. Z Evklidovim algoritmom izračunamo največji skupni delitelj števil *k* in *N.* Če rezultat izračuna ni enak 1, smo z algoritmom končali, saj je največji skupen delitelj eden od faktorjev števila *N*. Če rezultat ni enak 1, nadaljujemo z naslednjim korakom.
4. Definirajmo novo spremenljivko q = 1. Nato izračunamo $(q \cdot k)\ mod\ N$. Če rezultat ni enak 1, spremenljivki *q* dodelimo vrednost ostanka pri računanju. Ta postopek ponavljamo, dokler rezultat ni enak 1. Nato definiramo še spremenljivko *r* in ji dodelimo vrednost, ki je enaka številu transformacij, ki smo jih opravili v prejšnjem postopku.
5. V tem koraku preverimo vrednost spremenljivke *r*. Če $\exists x \in \mathbb{N},\ r = 2x + 1$, se vrnemo na 2. korak, vendar sedaj izberemo drugo vrednost *k.*
6. Če $\exists x \in \mathbb{N}$, da velja $r = 2x$, definiramo novo spremenljivko *p*, katere vrednost je enaka vrednosti transformacije *r/2*. V primeru, da je $p + 1 = N$, se vrnemo nazaj na 2. korak, izberemo novo poljubno vrednost *k* in postopek ponovimo.
7. V zadnjem koraku faktorje števila izračunamo kot $f_1 = D(p + 1, N)$ ter $f_2 = D(p - 1, N)$. Števila f_1 ter f_2 sta faktorja števila N. [19]

6.4 Odpornost eliptičnih krivulj na napad kvantnih računalnikov

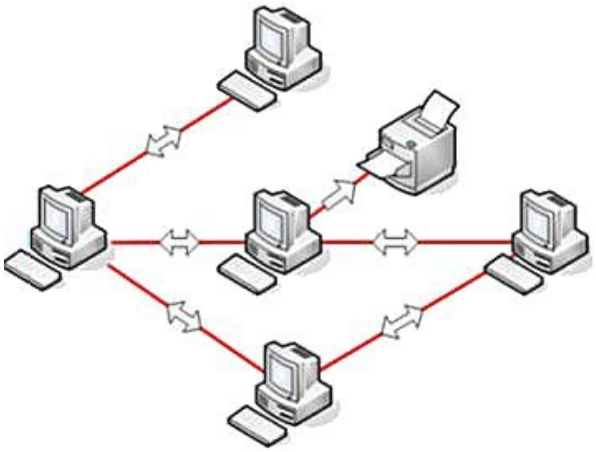
Mnoge strokovnjake s področja kriptografije skrbi, da ekstremno hiter razvoj kvantnih računalnikov ogroža današnje asimetrične kriptografske sisteme. Zlom asimetrične kriptografije bi spremenil način poslovanja, delovanja interneta, spletnega bančništva, …

V prejšnjem poglavju smo se poglobili v znani kvantni algoritem, t. i. Shorov algoritem. Kot smo izvedeli, s pomočjo algoritma ugotovimo prafaktorje poljubnega sestavljenega števila v kratkem postopku, ki ni pretežno težak. Shorjev algoritem je v teoriji najboljše orožje, ki ga lahko uporabimo proti našim trenutnim kriptografskim sistemom, saj ti temeljijo na

faktoriziranju števil, vendar to ne pomeni, da je lahko v praksi uporabljen danes. Shorjev algoritem je efektiven za faktoriziranje relativno malih števil, ampak današnji kriptografski sistemi ne uporabljajo majhnih števil, zato Shorov algoritem ni najboljša izbira za napad kriptografskega sistema. Pri eliptičnih krivuljah uporabljamo 256 bitne ključe, za katere bi Shorov algoritem potreboval preveč časa, torej je algoritem trenutno neefektiven. Da bi kvantni računalnih s Shorovim algoritmom razbil enkripcijo eliptičnih krivulj v razumno kratkem času, bi potrebovali kvantni računalnik z 2330 kubiti ter 126 miljardami Toffolijevih vrat. Zaenkrat tako zmogljivega kvantnega računalnika še nismo izumili, zato je trenutno to nemogoče izvesti, a v prihodnosti bo to definitivno mogoče. Največja nevarnost kriptografiji eliptičnih krivulj, ki jo danes prinašajo kvantni računalniki, je dolgoročno šifriranje dokumentov, saj bodo le-ti v prihodnosti nezavarovani.

7 RAČUNALNIŠKA OMREŽJA

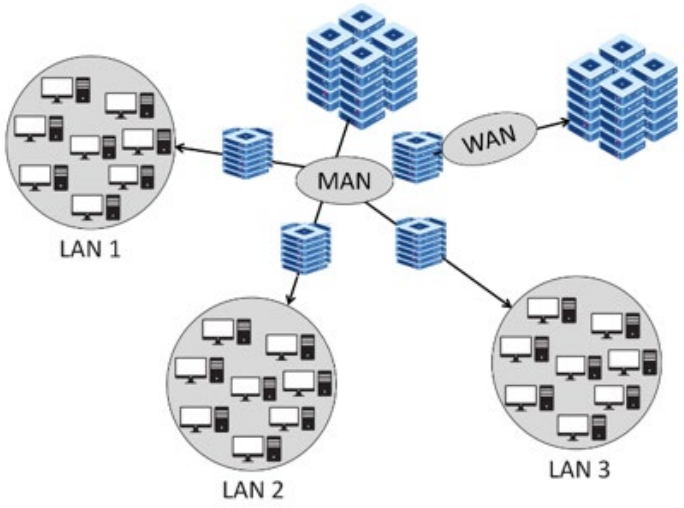
Računalnik je predstavljal revolucionarni izum, ki je človeštvo popeljal v novo digitalno dobo, saj je bila s tem omogočena hitra in učinkovita obdelava podatkov, kot tudi njihovo hranjenje. Kljub temu je še vedno obstajala potreba po hitrem prenosu informacij med različnimi računalnik, zato so računalničarji vzpostavili računalniška omrežja, ki omogočajo deljenje informacij in storitev med povezanimi računalniki. Računalniška omrežja lahko delimo glede na velikost, lokacijo, uporabljeno tehnologijo prenosa podatkov, arhitekturo itd.



Slika 7: Omrežje enakovrednih partnerjev [20]

7.1 Vrste omrežji

Omrežja najlažje delimo glede na velikost ter njihovo lokacijo. Povezovanje računalnikov v manjša omrežja omogoča obstoj svetovnega spleta, saj povezava vsakega računalnika neposredno v globalno omrežje ni izvedljiva, sploh iz vidika dodeljevanja unikatnih IP naslovov. Zatorej računalnike povezujemo v lokalna omrežja (LAN), ki najpogosteje pokrivajo dom, pisarno ali oddelek. Ta se nato s pomočjo usmerjevalnika povezujejo v večja, mestna omrežja, poznana tudi pod kratico MAN. Zaradi dandanašnje prepletenosti povezav računalnikov se mestna omrežja povezujejo v WAN, to so omrežja, ki pokrivajo celotne pokrajine, države in celo celine.

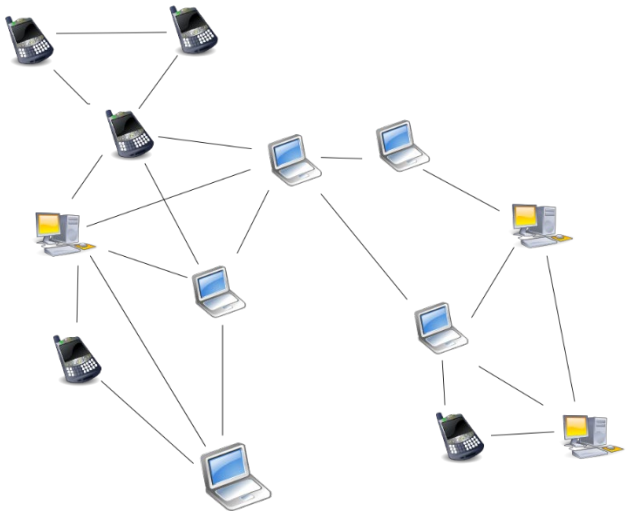


Slika 8: LAN, MAN in WAN [21]

7.2 Arhitektura omrežji enakovrednih partnerjev

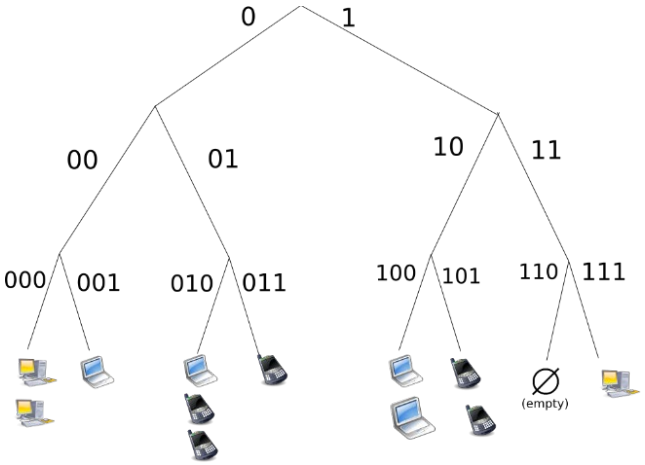
Omrežje enakovrednih partnerjev se za razliko od omrežji odjemalec-strežnik odpoveduje specializaciji vozlišč za določene namene, zato vsak računalnik v tem omrežju opravlja nalogo odjemalca, torej pošilja zahteve, kot tudi strežnika, torej se odziva na zahteve drugih. Zaradi pomanjkanja specializacije je omrežje težko optimizirati, zatorej je v sodobnem spletu redkeje uporabljeno, saj je učinkovito širjenje na več tisoč vozlišč skoraj neizvedljivo. Kljub prej naštetim pomanjkljivostim ima omrežje enakovrednih partnerjev prednost pri odpornosti pred izpadi. Če v omrežju odjemalec-strežnik odpove strežnik, komunikacija v tem omrežju ni več izvedljiva, saj je strežnik skrbel za usmerjanje prometa, kot se tudi odzival na zahteve, medtem ko je komunikacija v omrežju enakovrednih partnerjev mogoča, dokler sta v omrežju delujoča vsaj dva vozlišča in sta le-ta tudi povezana med sabo. S tem v takšnih omrežjih pridobimo večjo odpornost pred izpadi omrežji, kot pa tudi zmanjšamo odpornost pred cenzuro s strani večje entitete, zato so te vrste omrežji priljubljene predvsem pri prenašanju ilegalno pridobljenih multimedijskih vsebin in datotek. [22]

Omrežja enakovrednih partnerjev glede na način usmerjanja podatkovnih paketov v omrežju delimo na strukturirane, nestrukturirane in hibridne modele. Nestrukturirana omrežja so najlažja za vzpostavitev, saj pri povezovanju novih vozlišč v omrežje te ni potrebno vključiti v neko vnaprej določeno topologijo. V takšnih omrežjih vozlišča tvorijo naključne povezave s sosednjimi vozlišči, zatorej je prenos podatkov upočasnjen, saj nihče natančno ne pozna poti od izvora do cilja.



Slika 9: Nestrukturirano omrežje enakovrednih partnerjev [23]

Če želimo v omrežju enakovrednih partnerjev vzpostaviti učinkovitejši prenos podatkovnih paketov, je potrebno vzpostaviti ustrezno topologijo. Takšna omrežja običajno delujejo na podlagi dveh tabel, in sicer prve, kjer je zapisano katero vozlišče hrani določene podatke, ter druge, kjer so zapisane vse direktne povezave, ki jih je posamezno vozlišče vzpostavilo. S pomočjo teh dveh tabel se lahko vozlišča primerno odločajo o nadaljnji poti podatkovnih paketov skozi omrežje.



Slika 10: Strukturirano omrežje enakovrednih partnerjev [24]

7.3 Metode za komunikacijo med napravami

Neodvisno od uporabljenega tipa omrežja, naprave med seboj ne bodo mogle komunicirati, če nimajo pripisanega unikatnega imena. V računalništvu s tem namenom uporabljamo IP naslove. V verziji IPv4 naslov sestavljajo štiri 8-bitna števila. IP naslov vsakega računalnika v omrežju mora biti edinstven, saj lahko le tako z gotovostjo zagotovimo, da bodo podatki

vedno prispeli na željeno destinacijo. Poleg IP naslova je pomembna tudi uporaba omrežne maske, s pomočjo katere IP naslov ločimo na naslov omrežja in naslov naprave. V lokalnih omrežjih najpogosteje uporabimo omrežno masko 255.255.255.0, ki bi v primeru IP naslova 192.168.1.13, le-tega ločevala na omrežni naslov, 192.168,1.0, in naslov naprave, 0.0.0.13. [25]

Za uspešno pošiljanje podatkov skozi omrežje potrebujemo še protokol, ki usmerja podatke skozi omrežje. V današnjem času uporabljamo predvsem FTP in TCP. Funkcionalnost obeh je podobna, razlika se pokaže šele pri podrobnem ogledu načina pošiljanja paketov. FTP prične s pošiljanjem podatkov neodvisno od stanja prejemnika, torej pošilja podatke tudi neodzivnemu prejemniku, med tem ko se TCP s pomočjo metode trosmernega usklajevanja (*ang. three-way handshake*) pred pričetkom pošiljanja prepriča o uspešno vzpostavljeni povezavi s prejemnikom. Zaradi vnaprej vzpostavljene povezave TCP predstavlja zanesljivejši protokol za prenos čez omrežje, zato je tudi pogosteje uporabljen protokol transportne plasti. Ko želimo poslati podatke s pomočjo TCP, jih bo ta razdelil na manjše pakete podatkov in jih opremil z naslednjimi informacijami za sprejem: številka vrat, zaporedna številka paketa in kontrolna vsota. Prejemnik mora po sprejemu podatkovnih paketov na ustreznih vratih iz njih izluščiti podatke ter jih smiselno sestaviti nazaj v celoto. Pri tem si pomaga z zaporedno številko paketa, ki mu pove vrstni red podatkov in kontrolno vsoto, s katero prejemnik zagotovi celovitost prejetih podatkov. Ko prejemnik preveri pravilnost prejetih paketov, pošiljatelju odgovori z drugim podatkovnim paketom, v katerem mu sporoča pravilnost prejema. [26]

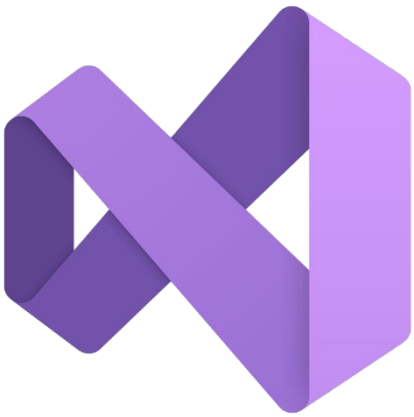
Brez protokola TCP in IP naslovov bi bila komunikacija v omrežjih močno otežena. Ker sta ta dva protokola za prenos podatkov v sodobnem spletu pogosto uporabljena skupaj, se v praksi pojavlja poimenovanje TCP/IP, ki se nanaša na sklad komunikacijskih protokolov.

8 PROGRAMSKA ORODJA

Med izdelavo kriptografske knjižnice in kasneje grafične aplikacije smo se posluževali predvsem dveh programskih orodij, in sicer Visual Studio 2022 ter GoLand. Ti dve programski orodji sta integrirani razvojni okolji (IDE), ki omogočata lažje pisanje, spreminjanje in testiranje napisane programske kode. Prav tako IDE pogosto vključuje prevajalnike uporabljenih programskih jezikov ter razhroščevalnike. Za lažje sodelovanje pri izdelavi programske kode smo uporabljali tudi GitHub.

8.1 Visual Studio 2022

Visual Studio 2022 je integrirano razvojno okolje, ki ga je razvilo podjetje Microsoft. Omogoča razvoj aplikacij v številnih podprtih programskih jezikih, npr. C#, C++, Python, Visual Basic, XML, … Poleg razvoja namiznih aplikacij omogoča razvoj spletnih strani, saj podpira programiranje s HTML, CSS, JavaScript ter PHP. Njegovo priljubljenost lahko pripišemo uporabi umetne inteligence IntelliSense, ki s pomočjo prepoznavanja konteksta programerju pomaga s pisanjem kode. Za uporabo Visual Studia 2022 pri izdelavi kriptografske knjižnice ter grafične aplikacije smo se odločili zaradi inteligentih predlogov med pisanjem programa, odličnega razhroščevalnika, dobre integracije s C# ter WPF in možnosti brezplačne uporabe tega integriranega razvojnega okolja. [27]



Slika 11: Visual Studio 2022 [28]

8.2 GoLand

GoLand je integrirano razvojno okolje razvito s strani podjetja JetBrains. Za razliko od Visual Studia, ki podpira številne programske jezike, je GoLand primerno zasnovana za pisanje in urejanje programske kode napisane v programskem jeziku Go. Kljub temu urejevalnik še

vedno podpira uporabo JavaScripta ter TypeScripta. Program je plačljiv za uporabo, vendar s svojim razhroščevalnikom in pomočjo pri pisanju kode v Go upraviči svojo ceno. GoLand omogoča tudi odlično integracijo z GitHubom, kar je nadvse pomembno, ko ima razvijalec opravka z večjimi projekti. [29]

Slika 12: GoLand [30]

8.3 GitHub

GitHub je eno izmed najbolj priljubljenih orodji za razvijalce programske opreme, saj omogoča preprost in efektiven način deljenja programske kode med več razvijalci, kot tudi skupno sodelovanje na projektih. Orodje je brezplačno za uporabo, za podjetja in zahtevnejše uporabnike pa ponuja tudi plačljive opcije, ki med drugim vključujejo več prostora za shranjevanje, večjo fleksibilnost projekta ter dodajanje hierarhije razvijalcev. Projekt je izredno uporaben tudi pri deljenju odprtokodnih projektov, saj je objava kode preprosta, prav tako pa vsem, ki si ogledujejo kodo, omogoča komentiranje kode, kot tudi podajanje izboljšane kode. [31]

Slika 13: Github [32]

9 PROGRAMSKI JEZIKI

Pred pričetkom izdelave uporabniške aplikacije smo se morali odločiti za primeren programski jezik, ki ga bomo uporabili pri izdelavi. Zaradi dobrega poznavanja programskega jezika, objektne usmerjenosti ter možnosti programiranja grafičnih aplikacij v WPF smo se odločili uporabiti C#. Po prvem testiranju aplikacije smo jo želeli še optimizirati, zato smo uporabili programski jezik Go, ki je sintaktično nezahteven, ponuja pa podobno hitrost delovanja kot denimo C in C++.

9.1 C#

C# je objektno usmerjen programski jezik razvit s strani podjetja Microsoft. C# si lahko predstavljamo kot naslednika programskih jezikov C, C++ ter Java, zato si ti jeziki delijo številne podobnosti. C# je bil prvič predstavljen javnosti leta 2000 z željo uvedbe preprostega, splošnega ter berljivega programskega jezika, ki kompleksnejše operacije izvede brez programerjeve izrecne definicije v kodi. Kombinacija teh značilnosti je na trgu očitno zaželena, saj je dandanes C# eden izmed 10 najbolj uporabljenih programskih jezikov. Poleg preproste uporabe se programska koda napisana v C# izvaja relativno hitro, saj se sprva prevede v nizkonivojski programski jezik, ta pa se tik pred izvedbo interpretira v strojno kodo določenega procesorja. [33]

Slika 14: C# [34]

9.2 Go

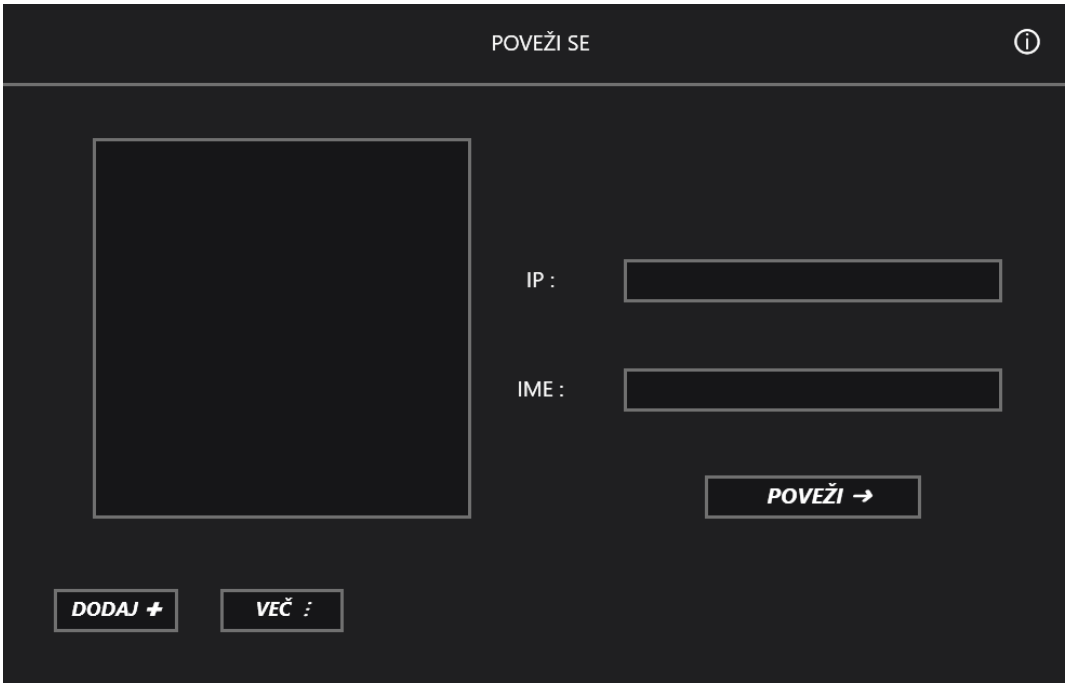
Go je odprtokodni programski jezik, ki ga je izdalo podjetje Google. Prvi razvijalci so bili Robert Griesemer, Rob Pike in Ken Thompson, ki so se lotili izdelave svojega programskega jezika, saj takrat na trgu ni obstajal programski jezik, ki bi ponujal preprosto sintakso, hkrati pa omogoča primerljivo hitrost izvajanja, kot jo ponujajo nizkonivojski programski jeziki. Hitrost izvajanja lahko pripišemo prevajanju programske kode neposredno v strojno kodo, kot tudi statično definiranje spremenljivk, kar pripomore k boljši optimizaciji porabe pomnilnika. Ker je Go veliko preprostejši za uporabo kot denimo C, hkrati pa ponuja primerljive hitrosti izvajanja, so razvijalci v programskem jeziku Go zelo iskani. [35]

Slika 15: Go [36]

10 PREDSTAVITEV APLIKACIJE

S pomočjo pridobljenega znanja o kriptografiji z eliptičnimi krivuljami in z uporabo znanja o programiranju v programskih jeziki C# ter Go smo se lotili izdelave uporabniške aplikacije za šifriranje in pošiljanje sporočil v P2P omrežju. Aplikacijo sestavljata dva različna uporabniška pogleda, ki sta podrobneje predstavljena v nadaljevanju.

Prvi uporabniški pogled, ki je viden na Sliki 16, je pogled, ki je uporabniku viden takoj ob zagonu aplikacije, in je namenjen vzpostavitvi povezave pred pričetkom klepeta. Prvi zaslon ponuja številne možnosti, ki olajšajo večkratno povezavo z isto osebo. Na levi strani zaslona je prikazan kvadrat, v katerem se bodo prikazali vsi shranjeni kontakti. Uporabniku tako ne bo potrebno večkrat vnašati IP naslova osebe, s katero je že komuniciral. Na desnem delu zaslona lahko vidimo obrazec, kamor vnesemo IP naslov in ime uporabnika, s katerim želimo komunicirati. Ko smo vnesli podatke, lahko izberemo gumb poveži, ki začne s postopkom vzpostavljanja povezave. Če uporabnik želi shraniti pravkar vnesene informacije v imenik, izbere gumb dodaj, levo spodaj. V primeru da ima uporabnik že shranjen kontakt, želi pa izvedeti njegov IP naslov ali urejati že shranjene podatke, v levem kvadratku izbere željenega uporabnika ter nato izbere gumb več. S tem se v desni obrazec izpišejo shranjeni podatki.



Slika 16: Zaslonska slika uporabniškega pogleda za vzpostavitev povezave

V primeru, da uporabnik ni seznanjen z uporabo te aplikacije oziroma mu del postopka za vzpostavitev povezave ni povsem jasen, lahko s klikom na gumb za dodatne informacije, desno zgoraj, odpre nov pogled, ki vsebuje navodila za uporabo, kot tudi kratko predstavitev delovanja aplikacije.

Ko je uporabnik vnesel podatke in izbral gumb poveži, bi se mu moral odpreti uporabniški pogled za klepet z izbranim uporabnikom, kot je razvidno na Sliki 17. V primeru da je uporabnik vnesel neveljaven IP naslov ali pa uporabnik ni na voljo, se bo na zaslonu izpisalo opozorilo, da povezave trenutno ni mogoče vzpostaviti.

V zgornjem delu zaslona se bodo prikazala vsa že poslana sporočila. Če uporabnik želi poslati novo sporočilo, ga najprej vnese v spodnji obrazec, nato pa mora izbrati gumb pošlji. Ko uporabnik pošlje sporočilo, se to s pomočjo izdelane knjižnice zašifrira in pošlje prek omrežja do prejemnika. Prejemnikova aplikacija s pomočjo izdelane knjižnice dešifrira sporočilo in ga uporabniku prikaže kot originalno sporočilo. Ob prejemu sporočila od sogovornika bo aplikacija predvajala zvok, ki uporabnika opozori na prejeto neprebrano sporočilo. Če uporabnik želi končati klepet, lahko izbere gumb končaj klepet, ki prekine povezavo in uporabnika vrne nazaj na prvi uporabniški pogled, kjer se lahko ponovno poveže s poljubnim uporabnikom. V primeru težav ima uporabnik možnost izbrati gumb za dodatne informacije, desno zgoraj, ki odpre nov pogled z navodili za uporabo.

Slika 17: Uporabniški pogled za klepet z uporabnikom

11 PREDSTAVITEV REZULTATOV RAZISKOVALNE NALOGE

Naša prva hipoteza trdi, da je problem diskretnega logaritma eliptičnih krivulj robustnejši, zatorej tudi odpornejši pred napadi, kot tisti znotraj grupe $\mathbb{Z}p^*$. Odpornost na napade najlažje ocenimo s primerjavo bitnih dolžin ključev, uporabljenih pri kriptografiji za dosego enake stopnjo varnosti. Dandanes kriptografija z eliptičnimi krivuljami uporablja 256 bitne ključe, kar je primerljivo s približno 15 300 bitimi ključi, ki bi jih morali uporabljati pri Diffie-Hellman izmenjavi ključev. Zaradi te ugotovitve lahko našo prvo hipotezo potrdimo.

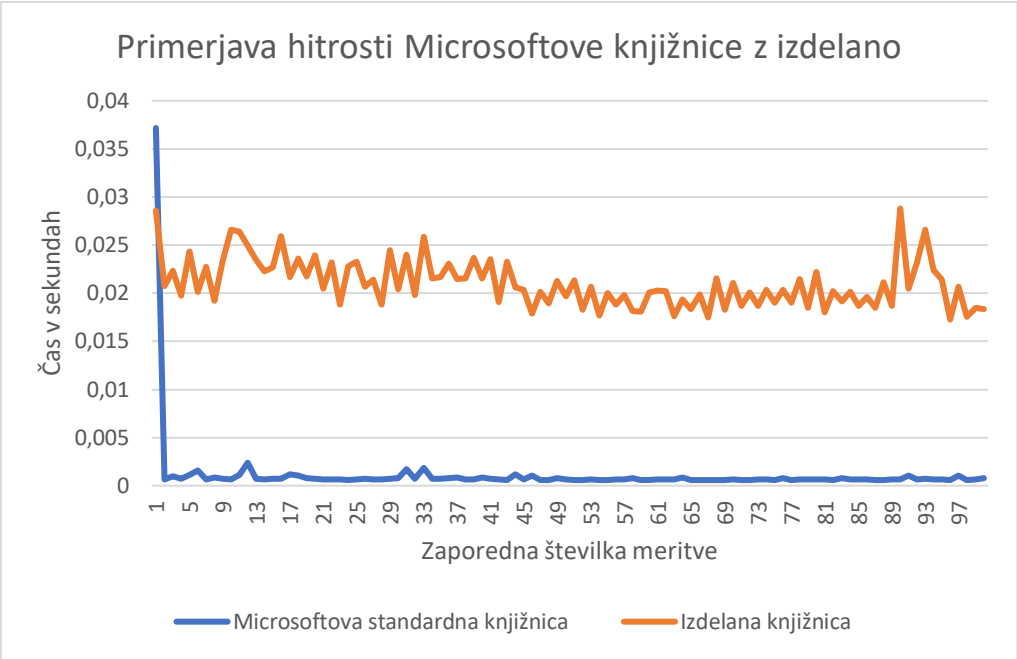
Druga hipoteza se je nanašala na odpornost kriptografije z eliptičnimi krivuljami pred napadi kvantnih računalnikov. V 6. poglavju smo ugotovili, da bi za trenutni zlom kriptografije potrebovali kvantne računalnike z več tisoč kubiti. V realnem svetu imajo raziskovalci na voljo kvantne računalnike, ki imajo nekaj sto kubitov, ti pa ne delujejo še povsem zanesljivo. Kdaj bodo kvantni računalniki zmožni zlomiti trenutno kriptografijo je težko napovedati, saj nekateri napovedujejo revolucijo v roku nekaj let, spet drugi menijo, da bo vzelo še vsaj nekaj desetletji razvoja. Na podlagi ugotovitev lahko drugo hipotezo ovržemo, saj kriptografija z eliptičnimi krivuljami ne bo odporna pred napadi kvantnih računalnikov, ko ti dosežejo primerno zmogljivost.

Tretja hipoteza se nanaša na izdelano aplikacijo, in sicer da je aplikacija primerna za vsakdanjo rabo. Med izdelavo raziskovalne naloge smo izdelali knjižnico za šifriranje z eliptičnimi krivuljami in jo kasneje povezali skupaj z uporabniškim vmesnikom izdelanim v WPF. S tem smo želeli približati izdelano aplikacijo laičnemu uporabniku, vendar uporaba aplikacije še vedno zahteva osnovno razumevanje delovanja računalniškega omrežja, saj mora uporabnik sprva vnesti IP naslov uporabnika, s katerim želi komunicirati. Z vidika funkcionalnosti aplikacija deluje brezhibno. Tako lahko tretjo hipotezo delno potrdimo, saj aplikacija omogoča vsakdanjo rabo, vendar zaradi zagotavljanja čim večje stopnje varnosti ni povsem prilagojena laiku.

S četrto hipotezo smo želeli primerjati hitrost in učinkovitost delovanja izdelane knjižnice za šifriranje s standardno Microsoftovo knjižnico. Da bi lahko čim bolje primerjali hitrosti delovanja, sta obe knjižnici napisani v istem programskem jeziku, testirani sta bili na isti strojni opremi, kot tudi vsako merjenje hitrosti je bilo izvedeno večkrat, s čimer smo se lahko izognili

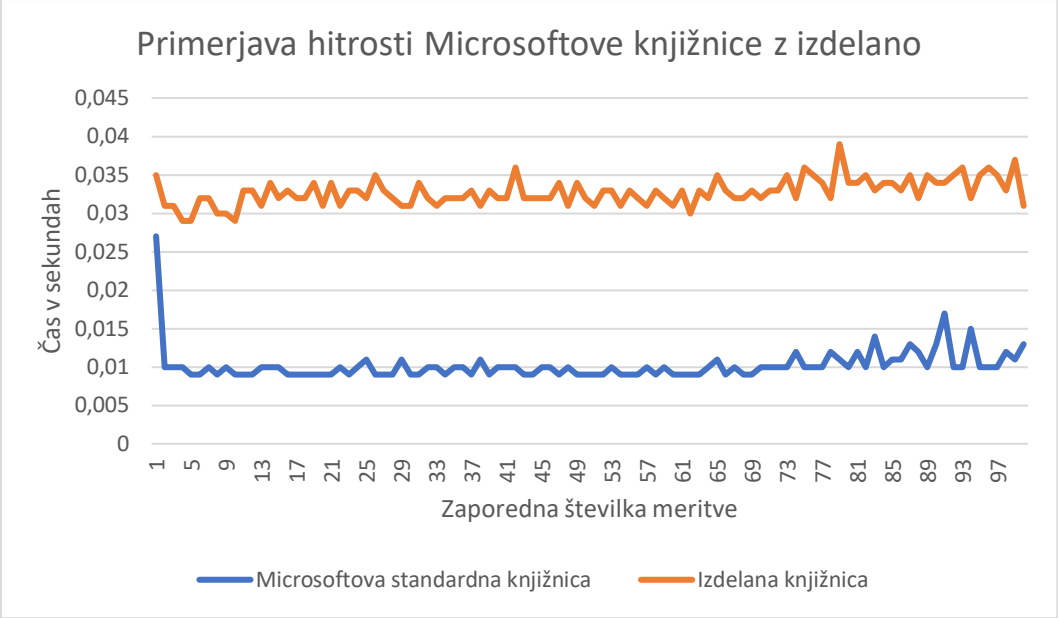
absolutnim in relativnim napakam med merjenjem. Po prvem merjenju rezultatov smo prišli do presenetljivih spoznanj. V nadaljevanju je predstavljanja podrobna analiza rezultatov.

Pri prvem merjenju smo se odločili primerjati hitrost izvajanja naše knjižnice v primerjavi z Microsoftovo. Da bi dosegli čim natančnejše rezultate, smo merjenje ponovili stokrat. Ugotovili smo, da je v prvem ciklu naša knjižnica hitrejša od Microsoftove, vendar ima po prvi izvedbi programa Microsoftova knjižnica veliko prednost pred nami, saj njihov čas izvajanja drastično pade, med tem ko se naša hitrost ne spreminja. Povprečen čas izvajanja naše knjižnice je 0,02103 sekunde, medtem ko ima Microsoftova knjižnica povprečni čas 0,00111 sekunde. Po prvem merjenju lahko zaključimo, da je njihova knjižnica približno 18-krat hitrejša od naše pri ponavljajočem se izvajanju. Natančnejši podatki so vidni na Grafu 1.



Graf 1: Primerjava hitrosti Microsoftove knjižnice z izdelano

Po prvem merjenju bi lahko zaključili, da je naša knjižnica veliko počasnejša, vendar nam ni bilo povsem jasno, zakaj je Microsoftova knjižnica veliko hitrejša od naše pri ponavljajočih se merjenih, vendar počasnejša pri enkratnem merjenju. Iz tega razloga smo se odločili ponoviti merjenja tako, da smo stokrat izmerili čas hitrosti delovanja, vendar smo za vsako iteracijo razred ponovno inicializirali. Z novimi podatki smo prišli do spoznanja, da je Microsoftova standardna knjižnica še vedno hitrejša, saj ima povprečni čas 0,01021 sekunde, medtem ko ima naša knjižica povprečno hitrost delovanja 0,03278, kar je še vedno 3-krat počasneje. Podrobnejša primerjava hitrosti je razvidna na Grafu 2.



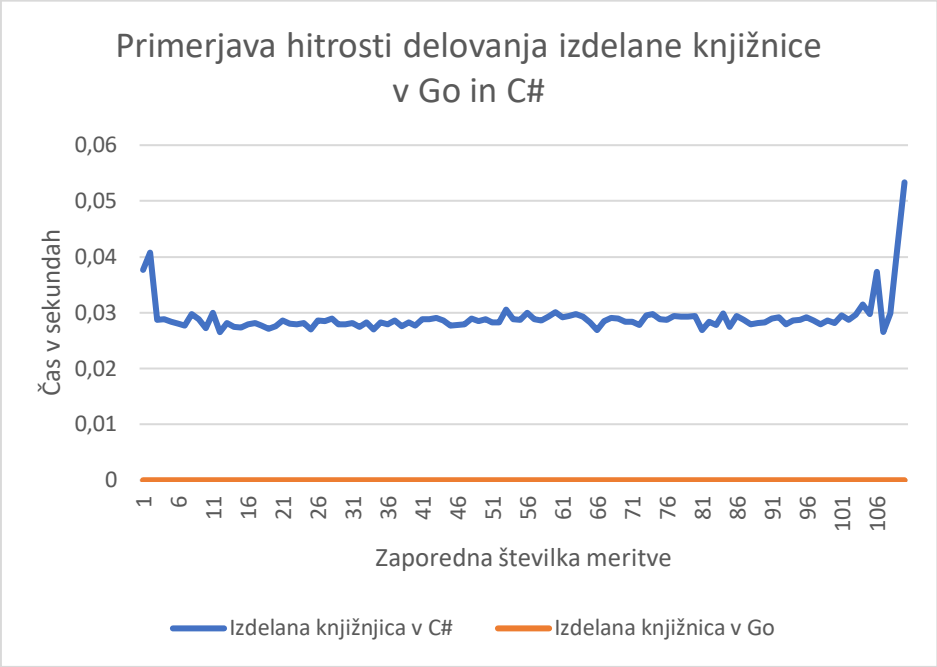
Graf 2: Primerjava hitrosti Microsoftove knjižnice z izdelano - ponovna inicializacija

Z obema meritvama smo prišli do naslednjih spoznanj:

- enkratna inicializacija razreda zniža povprečni čas izvajanja obeh knjižnic,
- večkratna inicializacija ima veliko manjši vpliv na hitrost delovanja naše knjižnice v primerjavi z Microsoftovo, čeprav je po hitrosti slednja še vedno hitrejša. Razliko pri hitrosti delovanja lahko najverjetneje pripišemo boljšemu upravljanju s pomnilnikom, predvsem shranjevanje v hitri spomin procesorja (ang. cache), in obravnavo velikih 256 bitnih števil v obliki bitov in ne s pomočjo podatkovnega tipa veliko celo število (ang. big integer). Po podrobnejši raziskavi smo ugotovili tudi, da je Microsoft za izdelavo svoje knjižnice uporabil programska jezika C in C++, za nekatere dele pa je uporabil celo Assembly, kar je znatno vplivalo na optimizacijo kode.

Ker je Microsoft z uporabo nižje nivojskih jezikov v svoji knjižnici dosegel znatno optimizacijo hitrosti delovanja, smo se odločili posnemati njihovo taktiko. Ker je razvoj v nižje nivojskih programskih jezikih zelo zahteven kot tudi zamuden, smo se odločili nastalo knjižnico prevesti v Go, ki je sodoben programski jezik s preprosto sintakso. Zaradi direktnega prevajanja izvorne kode v strojno dosega podobno hitrost izvajanja kot C oziroma C++. Z merjenjem smo ugotovili ogromno razliko v hitrosti delovanja. Koda napisana v Go se je izvedla v povprečno 120 nanosekundah, kar bi pomenilo 82 000-kratno razliko v primerjavi z Microsoftovo knjižnico in

kar 255 000-kratno razliko v primerjavi z našo knjižnico. Rezultati so podrobneje razvidni na Grafu 3.

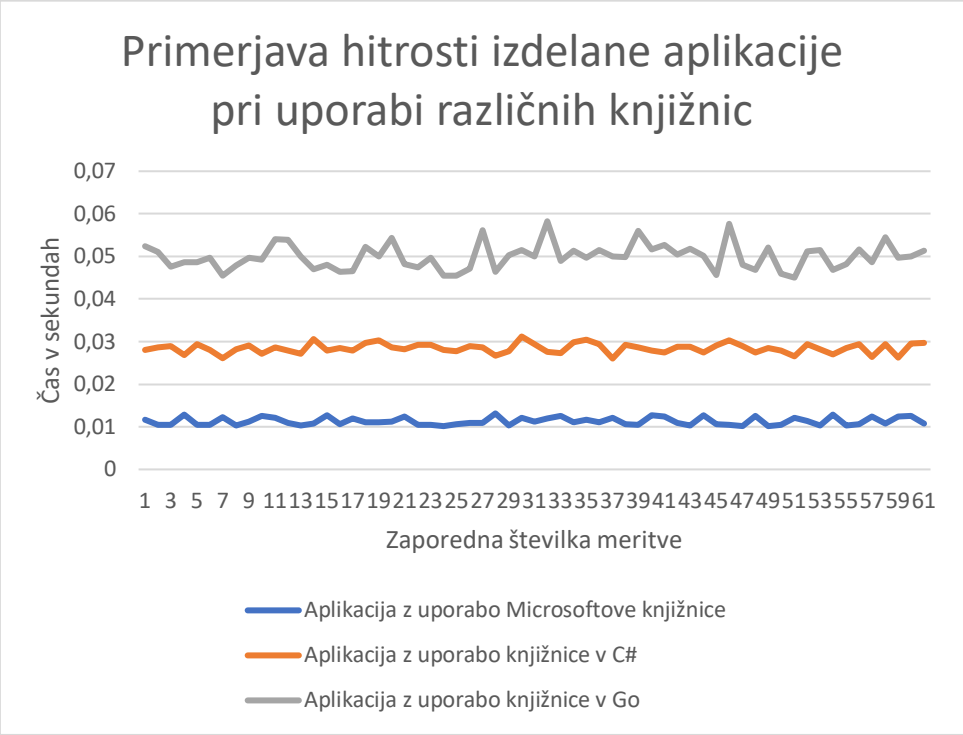


Graf 3: Primerjava hitrosti med izdelano knjižnico v C# in Go

Tako velika razlika v hitrosti delovanja med dvema programskima jezikoma se nam je sprva zdela nerealna, vendar smo jo pripisali temu, da sta kodi v C# napisani objektno, kar zaradi inicializacije objekta podaljša delovanje programske kode, medtem ko Go neposredno ne podpira objektno usmerjenega programiranja. Prav tako se vsaka koda napisana v C# izvaja v terminalu, ki zahteva določen čas, da se zažene. S pomočjo navedenih spoznanj smo prišli do zaključka, da več tisočkratna razlika v hitrosti delovanja ni povsem primerljiva, vendar ne smemo zanemariti dejstva, da je koda napisana v Go veliko hitrejša kot tista napisana v C#, zato smo se pri izdelavi uporabniške aplikacije odločili uporabiti kodo napisano v Go za računanje koordinat točk na krivulji.

Kljub obetavnim rezultatom pri hitrosti računanja točk v knjižnici izdelani v Go, smo pri povezovanju te knjižnice skupaj z uporabniškim vmesnikom naleteli na težavi pri hitrosti prenosa. Programska koda v C# je poslala zahteva po izračunu koordinat točk knjižnici napisani v Go, kjer je Go izvedel kalkulacijo ter jo vrnil C#. Ker direktna komunikacija med tema dvema jezikoma ni mogoča, smo kot vmesni jezik za komunikacijo med njima uporabili C. Kljub temu da C ponuja hitro izvajanje kode, je komunikacija med dvema jezikoma še vedno zelo počasna, zato so bili rezultati pri uporabi aplikacije z uporabo knjižnice napisani v Go

slabši od kode napisane le v C#. Na Grafu 4 je razvidna primerjava hitrosti aplikacije z uporabo Microsoftove knjižnice, izdelane knjižnice v C# in izdelane knjižnice v Go.



Graf 4: Primerjava hitrosti izdelane aplikacije pri uporabi različnih knjižnic

Našo zadnjo hipotezo o hitrosti delovanja lahko delno potrdimo, saj Microsoftova knjižnica deluje hitreje v primerjavi z našo implementacijo napisano v C#, hkrati pa je njihova knjižnica veliko počasnejša v primerjavi z izdelano knjižnico napisano v Go. Pri povezavi vseh treh knjižnic skupaj z uporabniško aplikacijo je po hitrosti še vedno najbolje uporabiti standardno Microsoftovo knjižnico.

12 ZAKLJUČEK

Z izdelavo raziskovalne naloge smo primarno želeli podrobno predstaviti delovanje asimetrične kriptografije z eliptičnimi krivuljami. Ker je področje kriptografije zelo obsežno, predvsem pa zahteva tudi dobro poznavanje matematičnih teorij, je sprva govora o kriptografskih algoritmih, grupah, Evklidovem algoritmu, modularni aritmetiki ter problemu diskretnega logaritma. V petem poglavju se je naloga osredotočila na povezavo pridobljenega splošnega matematičnega znanja s kriptografijo z eliptičnimi krivuljami. Nadaljnja poglavja so namenjena povezavi pridobljenega znanja o kriptografiji s potrebnimi znanji, s katerimi smo izdelali aplikacijo in kasneje potrdili oziroma ovrgli zadane hipoteze.

Ugotovili smo, da je kriptografija z eliptičnimi krivuljami veliko varnejša pred napadi kot kriptografija uporabljena pri izmenjavi ključev Diffie-Hellman, ElGamal, idr. Prišli smo tudi do spoznanja, da navkljub moči problema diskretnega logaritma, kriptografija z eliptičnimi krivuljami še vedno ne kljubuje moči kvantnih računalnikov, ki bodo v nekaj desetletjih popolnoma spremenili področje računalništva. S pomočjo pridobljenega znanja smo izdelali tudi uporabniku prijazno aplikacijo z grafičnim vmesnikom za pošiljanje šifriranih sporočil preko lokalnega P2P omrežja. Za potrebe delovanja uporabniške aplikacije je bila izdelana tudi kriptografska knjižnica, ki navkljub številnim poskusom optimizacije še vedno ne kljubuje Microsoftovi knjižnici za šifriranje z eliptičnimi krivuljami, zatorej menimo, da je bil pri izdelavi slednje knjižnice minimalno uporabljen C#, ampak so se posluževali nižje nivojskih programskih jezikov, to so Assembly, C in C++. Razlika v hitrosti je kar 18-kratna pri ponovni uporabi že inicializiranega razreda oziroma 3-kratna pri ponovni inicializaciji razreda za vsako iteracijo.

Izdelana raziskovalna naloga nam je prinesla številna nova spoznanja, ki jih bomo s pridom uporabili pri izdelavi naslednje raziskovalne naloge, najverjetneje s področja veriženja blokov.

13 VIRI IN LITERATURA

[1] M. Pogačnik, „Pregled kriptografije,“ v *Teorija Kriptologije*, Kranj, 2006, p. 3.

[2] Alliegiance, „Generate private encryption keys with the Diffie-Hellman key exchange,“ 20 september 2022. [Elektronski]. Available: https://null-byte.wonderhowto.com/how-to/generate-private-encryption-keys-with-diffie-hellman-key-exchange-0180269/.

[3] „Asimetrična kriptografija,“ 19 september 2022. [Elektronski]. Available: https://www.creaplus.com/sl/viri/blog/asimetricna-kriptografija-si.

[4] „Time-keeping on clock uses arithmetic modulo 12,“ [Elektronski]. Available: https://en.wikipedia.org/wiki/Modular_arithmetic#/media/File:Clock_group.svg. [Poskus dostopa 23 februar 2023].

[5] „Kongruenca,“ [Elektronski]. Available: https://sl.wikipedia.org/wiki/Kongruenca. [Poskus dostopa 14 september 2022].

[6] C. Paar, Understanding Cryptography, Berlin: Založba Spriger, 2010.

[7] T. Košir, „Evklidov algoritem,“ [Elektronski]. Available: https://www.fmf.uni-lj.si/~kosir/poucevanje/skripta/evklidalgoritem.pdf. [Poskus dostopa 21 september 2022].

[8] T. Abiy, „Extended Euclidean Algorithm,“ [Elektronski]. Available: https://brilliant.org/wiki/extended-euclidean-algorithm/. [Poskus dostopa 18 september 2022].

[9] „Razširjen Evklidov algoritem,“ 15 september 2022. [Elektronski]. Available: https://sl.wikipedia.org/wiki/Raz%C5%A1irjeni_Evklidov_algoritem.

[10] „Grupa,“ [Elektronski]. Available: https://www.znanjesveta.com/o/Grupa. [Poskus dostopa 15 september 2022].

[11] „Ciklične grupe,“ 18 september 2022. [Elektronski]. Available: https://osebje.famnit.upr.si/~penjic/algebraIII/zima2016_2017/03a%20Ciklicne%20grupe.pdf.

[12] B. Petelinek, „Diskretni logaritem,“ 16 oktober 2022. [Elektronski]. Available: https://dk.um.si/Dokument.php?id=14715&lang=slv.

[13] „Diffie-Hellman key exchange,“ [Elektronski]. Available: https://cryptography.fandom.com/wiki/Diffie%E2%80%93Hellman_key_exchange#Description. [Poskus dostopa 23 oktober 2022].

[14] A. Jurišič in J. Tonejc, „Pametne kartice 3. del: Napadi in obrambe: velike skrivnosti malih kartic,“ *Monitor,* pp. 44-51, 2001.

[15] K. Logar, 2019. [Elektronski]. Available: https://repozitorij.uni-lj.si/Dokument.php?id=120481&lang=slv. [Poskus dostopa 30 november 2022].

[16] J. Wohlwend, 30 september 2022. [Elektronski]. Available: https://math.mit.edu/~apost/courses/18.204-2016/18.204_Jeremy_Wohlwend_final_paper.pdf.

[17] E. Schaffer, „Intro to quantum computing: qubits, superposition and more,“ 19 avgust 2021. [Elektronski]. Available: https://www.educative.io/blog/intro-to-quantum-computing#quantum. [Poskus dostopa 15 februar 2023].

[18] G. Press, „27 milestones in the history of quantum computing,“ 18 maj 2021. [Elektronski]. Available: https://www.forbes.com/sites/gilpress/2021/05/18/27-milestones-in-the-history-of-quantum-computing/?sh=47f787dc7b23. [Poskus dostopa 16 februar 2023].

[19] K. Rakhade, „Shor's algorithm,“ 23 oktober 2020. [Elektronski]. Available: https://kaustubhrakhade.medium.com/shors-factoring-algorithm-94a0796a13b1. [Poskus dostopa 15 februar 2023].

[20] „Računališko omrežje - omrežje enakovrednih partnerjev,“ [Elektronski]. Available: http://colos.fri.uni-lj.si/eri/INFORMATIKA/RACUNALNISKA_OMREZJA/slike/peer-to-peer.jpg. [Poskus dostopa 12 februar 2023].

[21] Teachoo, „3 types of networks,“ 1 marec 2023. [Elektronski]. Available: https://www.teachoo.com/16678/3777/Question-2/category/Past-Year---3-Mark-Questions/.

[22] FRI, „Računalniško omrežje,“ [Elektronski]. Available: http://colos.fri.uni-lj.si/eri/INFORMATIKA/RACUNALNISKA_OMREZJA/RacunalniskoOmrezje.html. [Poskus dostopa 12 februar 2023].

[23] „Overlay network diagram for an unstructured P2P network,“ [Elektronski]. Available: https://en.wikipedia.org/wiki/Peer-to-peer#/media/File:Unstructured_peer-to-peer_network_diagram.png. [Poskus dostopa 20 februar 2023].

[24] „Structured P2P network diagram,“ [Elektronski]. Available: https://upload.wikimedia.org/wikipedia/commons/7/79/Structured_%28DHT%29_peer-to-peer_network_diagram.png. [Poskus dostopa 20 februar 2023].

[25] M. Cory, „Ip Address Definition: How It Works and Examples,“ 12 april 2022. [Elektronski]. Available: https://www.investopedia.com/terms/i/ip-address.asp. [Poskus dostopa 19 februar 2023].

[26] B. Lutkevich, „Transimission Control Protocol (TCP),“ 23 september 2021. [Elektronski]. Available: https://www.techtarget.com/searchnetworking/definition/TCP. [Poskus dostopa 19 februar 2023].

[27] „Visual Studio,“ [Elektronski]. Available: https://en.wikipedia.org/wiki/Visual_Studio. [Poskus dostopa 1 marec 2023].

[28] „Visual Studio 2022,“ [Elektronski]. Available: https://en.wikipedia.org/wiki/Visual_Studio#/media/File:Visual_Studio_Icon_2022.svg. [Poskus dostopa 1 marec 2023].

[29] „Go,“ [Elektronski]. Available: https://www.jetbrains.com/go/. [Poskus dostopa 1 marec 2023].

[30] „GoLand,“ [Elektronski]. Available: https://webobjects2.cdw.com/is/image/CDW/4912422?$product-detail$. [Poskus dostopa 2 marec 2023].

[31] J. Juviler, „What is GiHub?,“ 31 oktober 2022. [Elektronski]. Available: https://blog.hubspot.com/website/what-is-github-used-for?hubs_content=blog.hubspot.com%2Fwebsite%2Fwhat-is-github-used-for&hubs_content-cta=What%20is%20GitHub%3F. [Poskus dostopa 3 marec 2023].

[32] L. Giret, „GitHub is now being used by over 100 million developers,“ 26 januar 2023. [Elektronski]. Available: https://thurrott.s3.amazonaws.com/wp-content/uploads/sites/2/2023/01/GitHub.jpeg. [Poskus dostopa 3 marec 2023].

[33] R. Sheldon, „C#,“ [Elektronski]. Available: https://www.techtarget.com/whatis/definition/C-Sharp. [Poskus dostopa 4 marec 2023].

[34] A. Alvarez, „C# logo,“ 10 februar 2018. [Elektronski]. Available: https://upload.wikimedia.org/wikipedia/commons/4/4f/Csharp_Logo.png. [Poskus dostopa 4 marec 2023].

[35] C. Kolade, „What is Go?,“ 7 oktober 2021. [Elektronski]. Available: https://www.freecodecamp.org/news/what-is-go-programming-language/. [Poskus dostopa 4 marec 2023].

[36] GoAuthors, „Go Logo,“ 26 april 2018. [Elektronski]. Available: https://upload.wikimedia.org/wikipedia/commons/0/05/Go_Logo_Blue.svg. [Poskus dostopa 4 marec 2023].

[37] G. Jože, Elementarna teorija števil, Ljubljana: DMFA - založništvo, 2009.

[38] B. Ikenaga, „Cyclic groups,“ 18 september 2022. [Elektronski]. Available: https://sites.millersville.edu/bikenaga/abstract-algebra-1/cyclic-groups/cyclic-groups.pdf.

[39] „Kriptografija,“ 20 september 2022. [Elektronski]. Available: https://sl.wikipedia.org/wiki/Kriptografija.